(12)

Washington

WASHINGTON·UNIVERSITY·IN·ST·LOUIS

AD-A176 626

DTIC FILE COPY

# DELAY-DOPPLER RADAR IMAGING

Semi-Annual Progress Report No. 1
O.N.R. Contract N00014-86-K-0370
Period: 1 June 1986 - 30 November 1986



Principal Investigator: Donald L. Snyder



ELECTRONIC SYSTEMS
AND SIGNALS
RESEARCH LABORATORY

Department of Electrical Engineering
Campus Box 1127
One Brookings Drive
Washington University

**DELAY-DOPPLER IMAGING-RADAR**

**Semi-Annual Progress Report No. 1**

**Office of Naval Research Contract Number N00014-86-K-0370**

**Period Covered: 1 June 1986 - 30 November 1986**


**Principal Investigator:**

Donald L. Snyder
Director, Electronic Systems and Signals Research Laboratory
Washington University
Campus Box 1127
One Brookings Drive
St. Louis, Missouri 63130

**Scientific Program Officer:**

Dr. Rabinder Madan
Office of Naval Research
Code 1114SE
800 North Quincy Street
Arlington, Virginia 22217-5000

## DISTRIBUTION

| | copies |
|---|---|
| Mr. John W. Michalski<br>Office of Naval Research<br>Resident Representative<br>Federal Building, Room 286<br>536 South Clark Street<br>Chicago, Illinois 60605-1588 | 1 |
| Dr. Rabinder N. Madan<br>Office of Naval Research<br>800 N. Quincy Street<br>Code 1114SE<br>Arlington, VA 22217-5000 | 1 |
| Director<br>Naval Research Laboratory<br>Attn: Code 2627<br>Washington, DC 20375 | 1 |
| Defense Technical Information Center<br>Building 5<br>Cameron Station<br>Alexandria, VA 22314 | 12 |
| Mr. Harper J. Whitehouse<br>Naval Ocean Systems Center<br>San Diego, California 92152 | 1 |

CONTENTS

# I. INTRODUCTION

This first semi-annual progress report contains a summary of the problem that is being addressed in this project, a summary of the work that is now in progress, a list of personnel who are participating in this project, and a summary of project-related activities.

The goal of this project is to formulate and investigate new approaches for forming images of radar/sonar targets from spotlight-mode, delay-doppler measurements. Initially, we are studying a particular processing motivated by an approach used in radionuclide imaging. Our longer term goal is to develop new processing based upon a realistic model for the data acquired with a radar-imaging system.

Inverse synthetic-aperture imaging (ISAR) in radar and sonar relies upon the relative motion between the transmitter, target, and receiver. In the usual approach, the target is illuminated by a series of transmitted pulses. The return for each pulse is a superposition of reflections from various locations on the target, with each location affecting the pulse by introducing both a delay and doppler shift. The returns are processed to produce an image of the target.

The common approach is to use the same transmitted-pulse for each illumination of the target. Bernfeld [1] appears to be the first to introduce the idea for radar imaging of modifying the pulse shape on successive illuminations. We are using this idea of Bernfeld's. He also suggested an approach for processing the reflected return pulses so as to produce images of the target; his approach is based on an analogy he observed to the equations governing the data acquired in the x-ray tomographs currently being used for forming radiological images in medicine.

One deficiency in Bernfeld's novel approach is that, for the analogy to

x-ray tomography to be accurate, the ambiguity function of the transmitted pulses must be highly concentrated along lines in the delay-doppler coordinates and, moreover, must have a constant amplitude along such lines. Thus, practical radar/sonar pulses having ambiguity functions with complicated sidelobe structures and a nonuniform amplitude along the principle lobe are not accommodated very well in the concept. The purpose of the initial phase of our study is to investigate an analogy to medical imaging where this restriction is relaxed. This extension to Bernfeld's idea may permit improved images to be formed for practical ambiguity functions.

The analogy to medical imaging which we are attempting to exploit is described fully in our paper [2], which is included herewith as Appendix 1. To summarize, the transmitted pulse has a complex envelope $E_t^{1/2}f(t)$, where $E_t$ is the transmitted energy, and the received pulse has a complex envelope s(t) given by

$$s(t) = \int_{-\infty}^{\infty} b(t-\tau/2,\tau)E_t^{1/2}f(\tau)d\tau,$$

where $b(t,\tau)$ is a zero-mean, complex-valued Gaussian process modeling a diffuse reflection interaction at the target; $b(t,\tau)$ is the instantaneous strength of the reflection at time t and two-way delay $\tau$. In our initial study, we are assuming that the scatter process is stationary temporally and uncorrelated spatially (i.e., a WSSUS model in the terminology of Van Trees [3].) The power spectrum of $b(t,\tau)$ at a given delay $\tau$ is the target's scattering function $\sigma(\tau,f)$. For two distinct delays, say $\tau_1$ and $\tau_2$, the processes $b(t,\tau_1)$ and $b(t,\tau_2)$ are uncorrelated. For our initial study, the received pulse is first processed by a collection of bandpass matched filters and square-law envelope detectors, as shown in [2, Figure 1, see Appendix 1]; each bandpass matched filter is matched to a doppler-shifted

version of the transmitted pulse. This is a different form of front-end processing from the usual ISAR processing where two-dimensional Fourier transforms are used. Our motivations for the use of the BPMF-SLED are: 1, the BPMF has known qualities for suppressing the effects of additive noise; and 2, the BPMF-SLED receiver arises in a fundamental way for estimating delay and doppler [3]. Without additive noise, the expected value of the result of this BPMF-SLED preprocessing is a function $p(\tau,f)$ of delay $\tau$ and doppler $f$ given by

$$p(\tau,f) = E_t\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\sigma(\tau',f')a(\tau-\tau',f-f')d\tau'df',$$

where $a(\tau,f)$ is the ambiguity function of the transmitted pulse, which is the squared magnitude of the complex delay-doppler correlation function [3, eqn (10.18)]. We call $p(\tau,f)$ the ''delay-doppler power function;'' it is the two-dimensional convolution of the ambiguity function of the transmitted pulse and the scattering function of the target.

Recognizing the effect on the ambiguity function of varying the linear FM sweep (chirp) rate of the signal is important for our work. The effect is to shear or tilt the ambiguity function in the delay-doppler plane [3, p. 290]. We denote this tilt by the parameter $\Theta$, which depends on the chirp rate. While it is not necessary to do so, we are assuming in our initial studies that the pulse shape is changed along with the the chirp rate in such a way that the only variation of the ambiguity function is to rotate it to an angle $\Theta$ in the delay-doppler plane; our motivation for this is to maintain a close analogy to radionuclide imaging. In order to include this chirp-rate modulation in our notation, we shall replace $f(t)$, $p(\tau,f)$, $s(t)$, and $a(\tau,f)$ by $f_\Theta(t)$, $p_\Theta(\tau,f)$, $s_\Theta(t)$, and $a_\Theta(\tau,f)$ respective-

ly.

We may now state the delay-doppler radar imaging problem that we are studying as follows. Estimate the target's scattering function from data acquired from a series of target illuminations for chirp rates resulting in N tilt angles $\theta_1, \theta_2, \ldots, \theta_N$ of the ambiguity function of the transmitted pulse. For our initial study in which the received signal is preprocessed by a collection of BPMF-SLED's, the estimate is to be formed by using an appropriate algorithm on the N delay-doppler power functions

$$p_\theta(\tau,f) = E_t\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\sigma(\tau',f')a_\theta(\tau-\tau',f-f')d\tau'df',$$

(for $\theta = \theta_1, \theta_2, \ldots, \theta_N$) appearing as the output of the BPMF-SLED's.

Our approach is based upon the use of an algorithm, called the confidence-weighted (CW) algorithm, that is used in radionuclide imaging where an expression similar to that for the delay-doppler power functions $p_\theta(\tau,f)$ is obtained. With reference to [4], in radionuclide imaging, $p_\theta(\tau,f)$ corresponds to the intensity of detected coincidences, at the various observation angles, $\sigma(\tau,f)$ corresponds to the intensity of annihilations, and $a(\tau,f)$ corresponds to the measurement-error density, and the goal is to estimate the intensity of annihilations from observations of coincidences when the error density is known from calibration experiments. The steps required with this algorithm are detailed in [2, see Appendix 1].

Our plan for studying the use of the CW algorithm for radar imaging consists of three major steps as follows:

1. establish a capability for simulating noise-free return signals from simple radar targets, simulating the BPMF-SLED

preprocessing, simulating the CW algorithm, simulating the conventional ISAR algorithm, and comparing results;

2. establish a capability for simulating noisy return signals from simple radar targets, simulating the BPMF-SLED preprocessing, simulating the CW algorithm, simulating the conventional ISAR algorithm, and comparing results;
3. formulate a model for the noisy return signal in an imaging radar system and apply statistical estimation theory to derive a model-based algorithm for forming the radar image, and compare this to the results obtained with the CW and conventional ISAR algorithms.

Our effort during the first six months has been toward accomplishing the first of these tasks.

## II. SUMMARY OF WORK ACCOMPLISHED

The goal of our efforts during the first six months of this project has been to establish a capability to simulate idealized delay-doppler radar data and to process the simulated data using both conventional ISAR processing and the CW processing suggested by that used in radionuclide imaging, as discussed in [2, see Appendix 1]. The implementation is ''idealized'' in the sense that no noise or random-reflection effects are included.

During this period, computer simulations of radar returns from simple targets and the processing of the returns by radar image-processing algorithms have been developed, and some preliminary tests have been performed.

Two targets have been simulated, a rotating rough disk and a rotating

rough sphere. The scattering functions for these test objects are known analytically and are shown in [2, Figures 2 and 5, see Appendix 1].

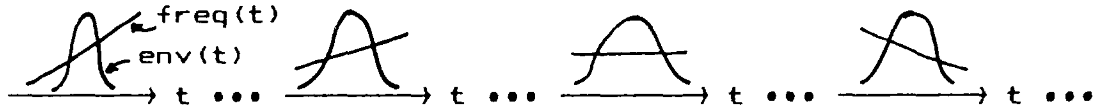
Two types of radar pulse-sequences have been included. They are illustrated in Figure 1. For the first, a sequence of 128 linear FM chirped pulses, each having a Gaussian envelope, is transmitted, with each pulse having a distinct chirp rate. The duration and FM sweep rate of each pulse were chosen so that the ambiguity functions associated with the resulting pulses would all have the same shape in the delay-doppler plane but would have their major axes rotated from one pulse to the next so as to cover all angles in the range from 0 to 180 degrees uniformly. The parameters for this first pulse sequence are given in Table 1.

For the second series of radar pulses simulated, a sequence of 128 stepped-frequency bursts is transmitted, with each burst consisting of 128 short, separated pulses; each pulse in a burst has a distinct frequency, but all the bursts are identical. Our motivation for using the stepped-frequency waveform is that it is one used at the Naval Ocean Systems Center in San Diego. The parameters for this second pulse sequence are specified in Table 2.

Two radar signal-processing algorithms have been implemented during this six month period. The first is based on conventional ISAR processing, and the second is based on the processing used in radionuclide imaging.

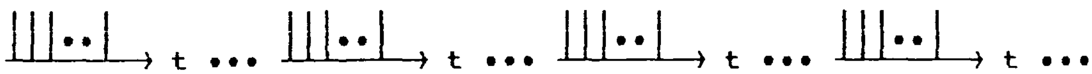
<u>processing based on conventional ISAR techniques</u>. A digital simulation capability was developed for generating ISAR images. This involved a study of current ISAR processing techniques and the selection of one approach for implementation. The literature (see, for example, D. L. Mensa [5]) describes ISAR digital processing in terms of two-dimensional discrete Fourier transformation techniques; this

SEQUENCE 1: GAUSSIAN ENVELOPE, LINEAR FM CHIRP, CHIRP-RATE MODULATION




| | | | | |
|---|---|---|---|---|
| PULSE #: | 1 | 32 | 64 | 96 |
| FWHM (µs): | 69.8 | 344 | 482 | 344 |
| CHIRP RATE (MHz/s): | +96 | +80 | 0 | -80 |
| BAND WIDTH (kHz): | 26 | 57.8 | 1.8 | 57.8 |
| AMBIGUITY FNC TILT ANGLE (θ): | 1.4° | 45° | 90° | 135° |

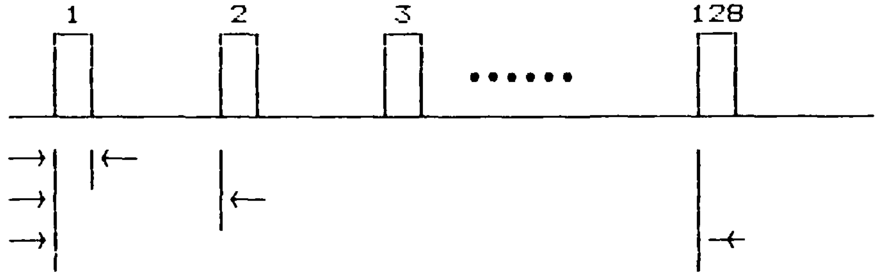
SEQUENCE 2: STEPPED-FREQUENCY WAVEFORM




| | | | | |
|---|---|---|---|---|
| BURST #: | 1 | 32 | 64 | 96 |
| CHIRP RATE (MHz/s): | 3.4 | 3.4 | 3.4 | 3.4 |

BURST PARAMETERS:

PULSE #:




T1 = 85 ns
T2 = 577 µs
T3 = 73 ms

FIGURE 1. SEQUENCES USED IN SIMULATIONS

TABLE 1: PARAMETERS OF SIMULATIONS USING A GAUSSIAN PULSE SEQUENCE

| QUANTITY | SYMBOL | VALUE |
|---|---|---|
| target distance | R | 12.5 nautical miles |
| number of pulses | N | 128 |
| center frequency | fc | 3 GHz |
| wavelength | $\lambda$ | 0.1 m |
| speed of light | c | |
| target rotation rate | $\omega$ | |
| down range resolution | rd | |
| cross range resolution | rc | |
| (maj axis)/(min axis) of ambiguity function | e | |
| target diameter | D | 64*rd |
| nominal two-way delay | Tr | 2*R/c |
| delay resolution | rt | 2*rd/c |
| doppler resolution | rf | $2*\omega*rc/\lambda$ |
| angle major axis amb. fnc. makes with the delay axis, pulse i | $\theta$ | $(i-1)/n\pi$ |
| constant | a1 | $2\pi e$ |
| constant | a2 | $2\pi/e$ |
| parameter | p1 | $a1*\sin^2(\theta) + a2*\cos^2(\theta)$ |
| parameter | p2 | $(a2-a1)*\sin(2\theta)$ |
| pulse std. deviation | T | $\mathrm{sqrt}(p1*rt/rf)/2\pi$ |
| pulse duration (FWHM) | FWHM | $2.355*(1+\lvert b\rvert T^2)/\pi T)$ |
| FM sweep rate | b | $p2/(8\pi T^2)$ |
| pulse bandwidth | BW | $(1+bT^2)/T$ |
| pulse spacing | T2 | |
| total illumination time | Tt * | sum[6*FWHM(i)] + (N-1)T2 |
| total aspect change | W | $\omega Tt$ |

EXAMPLE USED IN SIMULATION:

$rc = rd = 0.6$ m, $e = 7$, $\omega = 8.88\times10^{-3}$ rad/s

| QUANTITY | SYMBOL | VALUE |
|---|---|---|
| target diameter | D | 38.4 m |
| nominal two-way delay | Tr | 154 μs |
| pulse spacing | T2 | 577 μs |
| delay resolution | rt | 4 ns |
| doppler resolution | rf | 0.1066 Hz |
| constant | a1 | $14\pi$ |
| constant | a2 | $2\pi/7$ |
| min FWHM pulse duration | Tmin | 29.2 μs |
| max FWHM pulse duration | Tmax | 204.5 μs |
| min FM sweep rate | bmin | 0.0 Hz/s |
| max FM sweep rate | bmax | +/- 287 MHz/s |
| max pulse bandwidth | BWmax | 14.7 kHz |
| total illumination time | Tt | 316 ms * |
| * assumes no duty-cycle limitation on transmitter | | |
| total aspect change | W | 0.161 degrees |

## TABLE 2: PARAMETERS OF SIMULATIONS USING A STEPPED FREQUENCY SEQUENCE

| QUANTITY | SYMBOL | VALUE |
|---|---|---|
| target distance | R | 12.5 nautical miles |
| number of bursts | N | 128 |
| number of pulses/burst | n | 128 |
| center frequency | fc | 3 GHz |
| wavelength | λ | 0.1 m |
| speed of light | c | |
| target rotation rate | ω | |
| down range resolution | rd | |
| cross range resolution | rc | |
| target diameter | D | 64*rd |
| bandwidth | B | c/(2*rd) |
| burst duration | T3 | λ/(2*N*rc*ω) |
| pulse spacing | T2 | T3/(n-1) |
| pulse duration | T1 | 4*D/c |
| nominal two-way delay | Tr | 2*R/c |
| total illumination time | Tt | N*T3 |
| total aspect change | W | ω*Tt |

EXAMPLE USED IN SIMULATION:

rc = rd = 0.6 m, e = 7, ω = $8.88 \times 10^{-3}$ rad/s

| | | |
|---|---|---|
| target diameter | D | 38.4 m |
| bandwidth | B | 0.25 GHz |
| burst duration | T3 | 73 ms |
| pulse spacing | T2 | 577 μs |
| pulse duration | T1 | 85 ns |
| nominal two-way delay | Tr | 154 μs |
| total illumination time | Tt | 9.27 s |
| total aspect change | W | 4.7 degrees |

is the approach that has been adopted.

processing based on radionuclide imaging. A digital simulation capability was developed for generating delay-doppler power functions for the simple radar targets and Gaussian-envelope radar pulses described and for processing these power functions using the CW algorithm from radionuclide imaging.

Listings of the computer programs that have been developed to conduct these simulations are contained in Appendix 2.

Some of our first results obtained using the simulation capability that has been developed are shown in [2, Figures 2 to 6, see Appendix 1]. The reconstructions of the scattering functions of the two test targets obtained with our implementation of both processing approaches look quite close to the actual scattering functions. This has helped to verify the correctness of our implementations. The conventional ISAR processing does have sidelobe artifacts, with substantial power, that are not present with the new approach we are studying. These may be in part due to the fact that we have not yet introduced any windowing into the algorithm in order to maintain the greatest resolution. Windowing is now being introduced.

It is too early in our efforts to draw any major conclusions, but a tentative one for the simulations we have implemented thus far is that conventional ISAR processing and CW processing produce similar results when no additive noise or multiplicative randomness are present. We are encouraged by this because it indicates that CW processing can work. Our expectation is that when we introduce additive noise and a random reflection process, the CW algorithm will outperform the conventional ISAR processing both because of the use of the BPMF-SLED preprocessing and the fact that the processing used in radionuclide imaging was derived with

noise taken into consideration.

Activities preliminary to the formulation of an optical ISAR imaging problem have begun. Investigation to date (by K. Krause) is seeking to determine the feasibility of extending the concepts employed at microwave frequencies to the optical domain. This determination of feasibility has a number of interesting conceptual aspects that make the problem distinct. For example, the potential sources of image degradation, such as atmospheric turbulence, and the particular effects seen in optical radar images, such as speckle, need to be considered.

## III. WORK PLANNED

Additional work remains to complete the first task outlined above in the Introduction. Window functions need to be introduced into the conventional ISAR processing. The CW algorithm needs to be used for the stepped-frequency waveform, which requires an evaluation of the requisite ambiguity function. And more comparison studies are needed to reach firm conclusions about the relative performance of the ISAR and CW algorithms. This work is in progress.

We have initiated an effort to incorporate additive noise and a random reflection process into our simulation of radar-return data. When this is accomplished, we will process the simulated data using both the conventional ISAR algorithm and the CW algorithm in order to compare results.

We have begun to formulate the radar-imaging problem as one of statistical estimation. We have begun to examine the application of our approach to laser-radar imaging.

# IV. PERSONNEL

The following individuals have joined this research project during the last six months and attend regularly held group meetings to discuss it: R. C. Lewis, K. E. Krause, J. O'Sullivan, and J. T. Wohlschlaeger. Some brief biographical information on each follows.

NAME: Robert C. Lewis

PERSONAL INFORMATION:
Birthdate: June 22, 1955
Birthplace: Memphis, Tennessee

EDUCATION:
BSEE Memphis State University, 1978
MSEE Washington University, in progress

EMPLOYMENT:
McDonnell Douglas Corporation, St. Louis, 1978 to present
- contributed to the research and development in avionics of several advanced aerospace projects including the F-15 fighter and the Tomahawk cruise missile

PUBLICATIONS:

D. L. Snyder, H. J. Whitehouse, J. T. Wohlschlaeger, and R. C. Lewis, ''A New Approach to Radar/Sonar Imaging,'' Proc. 1986 SPIE Conf. on Advanced Algorithms and Architectures, Vol. 696, San Diego, CA, August 1986.

NAME: Kenneth E. Krause

PERSONAL INFORMATION:
Birthdate: November 3, 1951
Birthplace: Decatur, Illinois

EDUCATION:
BSEE University of Illinois, Urbana, IL, 1973
MSEE Bradley University, Peoria, IL, 1983
DSc EE Washington University, in progress

additional coursework in Physics, Math., and Chem.:
Illinois State University, Normal, IL (30 hours completed) 1979

**HONORS AND AWARDS**

Phi Kappa Phi

**EMPLOYMENT:**

General Telephone of Illinois, June 1973 - October 1979
- special service circuit equipment and transmission level specification

Caterpillar Tractor Co., October 1979 - June 1983
- electrical standards, computer numerical-control troubleshooting

McDonnell Aircraft Co., June 1983 - present
- mathematical modeling and computer analysis of radiative transfer processes in the IR and visible bands; visual response modeling

**NAME:** Joseph A. O'Sullivan

**PERSONAL INFORMATION:**

Birthdate: January 7, 1960
Birthplace: St. Louis, Mo.

**EDUCATION:**

| | |
|---|---|
| BSEE | University of Notre Dame, South Bend, In., 1982 |
| MSEE | University of Notre Dame, South Bend, In., 1984 |
| PhD EE | University of Notre Dame, South Bend, In., 1986 |

**EMPLOYMENT:**

Aug 1986 - present: EE Dep't, Washington University, Visiting Assistant Professor
Aug 1985 - May 1986: Univ. of Notre Dame, Senior Teaching Fellow
Jan 1982 - Aug 1984: Univ. of Notre Dame, Research Assistant
May 1982 - Aug 1982: Abacas Controls, Inc., New Jersey, Electrical Engineer

**PROFESSIONAL AFFILIATIONS AND HONORS:**

IEEE, Eta Kappa Nu, Burns Fellow

**PUBLICATIONS:**

J. A. O'Sullivan and M. K. Sain, ''A Theorem on the Feedback Equivalence of Nonlinear Systems Using Tensor Analysis,'' 23rd Allerton Conf. on Communication, Control, and Computing, U. of Illinois, Urbana, 1985.

J. A. O'Sullivan and M. K. Sain, ''Nonlinear Optimal Control with Tensors: Some Computational Issues,'' 1985 American Control Conf., Boston, MA.

J. A. O'Sullivan and M. K. Sain, ''Nonlinear Feedback Design: Optimal Responses by Tensor Analysis,'' 22nd Allerton Conf. on Communication, Control, and Computing, U. of Illinois, Urbana, 1984.

NAME: J. Trent Wohlschlaeger

PERSONAL INFORMATION:

birthdate: August 12, 1960
birthplace: Memphis, TN
resident of: Mesquite, TX

EDUCATION:

| | |
|---|---|
| BA | Math and Physics, Austin College, 1982 |
| BSEE | Washington University, 1985 |
| BSSSE | Washington University, 1985 |
| MSSSM | Washington University, 1985 |
| MSEE | Washington University, 1985 |
| Ph.D. EE | Washington University, in progress |

HONORS AND AWARDS:

Class Valedictorian, Noth Mesquite H.S.
National Merit Scholar
Member of Honorable Mention Team, 1985 Math Competition in Modeling
Sigma Pi Sigma, Tau Beta Pi, Eta Kappa Nu

EMPLOYMENT:

Amoco Research Center, Tulsa, OK, Geophysical Research Scientist, Summers 1982-1985

PUBLICATIONS:

D. L. Snyder, H. J. Whitehouse, J. T. Wohlschlaeger, and R. C. Lewis, ''A New Approach to Radar/Sonar Imaging,'' Proc. 1986 SPIE Conf. on Advanced Algorithms and Architectures, Vol. 696, San Diego, CA, August 1986.

## V. RELATED PROJECT ACTIVITIES

One paper has been published during the last six months. A reprint is included in Appendix 1.

R. C. Lewis, M. I. Miller, D. L. Snyder, and J. T. Wohlschlaeger visited the Naval Ocean Systems Center in August 1986 to discuss this project. M. I. Miller is a member of the Electrical Engineering faculty at Washington University. The visit was hosted by Mr. Harper J. Whitehouse of

NOSC. D. L. Snyder gave a seminar detailing the ideas and goals of the project for interested NOSC personnel. Discussions with several NOSC individuals interested in radar imaging took place during the day. A tour of the NOSC radar imaging facility concluded the visit.

K. E. Krause, D. L. Snyder, and J. T. Wohlschlaeger visited the Environmental Research Institute of Michigan (ERIM) in Ann Arbor, MI on October 7, 1986. The visit was hosted by Dr. James Fienup of ERIM. D. L. Snyder gave a seminar detailing the ideas and goals of the project for interested ERIM personnel, and discussions with several individuals interested in radar imaging took place.

In October 1986, D. L. Snyder gave a seminar about this project in the Department of Electrical Engineering at Washington University in St. Louis.

## VI. REFERENCES


1. M. Bernfeld, ''CHIRP Doppler Radar,'' Proc. IEEE, Vol. 72, No. 4, pp. 540-541, April 1984.

2. D. L. Snyder, H. J. Whitehouse, J. T. Wohlschlaeger, and R. C. Lewis, ''A New Approach to Radar/Sonar Imaging,'' Proc. 1986 SPIE Conference on Advanced Algorithms and Architectures, Vol. 696, San Diego, CA. (See Appendix 1.)

3. H. L. Van Trees, Detection, Estimation and Modulation Theory: Part III, John Wiley and Sons, 1971.

4. D. L. Snyder, L. J. Thomas, Jr., and M. M. TerPogossian, ''A Mathematical Model for Positron Emission Tomography Systems Having Time-of-Flight Measurements,'' IEEE Trans. on Nuclear Science, Vol. NS-28, pp. 3575-3583, June 1981.

5. D. L. Mensa, High Resolution Radar Imaging,'' Artech House, Inc., Dedham, MA, 1981.

# APPENDIX 1.

**Reprint of:**

D. L. Snyder, H. J. Whitehouse, J. T. Wohlschlaeger, and R. C. Lewis, ''A New Approach to Radar/Sonar Imaging,'' Proc. 1986 SPIE Conference on Advanced Algorithms and Architectures, Vol. 696, San Diego, CA., August 1986.

Presented at the 1986 SPIE Conference on Advanced Algorithms and Architectures, SPIE Proceedings Vol. 696, San Diego, CA.

# A NEW APPROACH TO RADAR/SONAR IMAGING[+]

Donald L. Snyder*, Harper J. Whitehouse**, J. Trent Wohlschlaeger*, and Robert C. Lewis*

* Washington University, St. Louis, MO 63130
** Naval Ocean Systems Center, San Diego, CA 92152

**Abstract**

We describe an analogy between imaging in delay-doppler radar/sonar and positron-emission tomography. This suggests new processing algorithms for the radar/sonar imaging problem that may permit improved visualization of targets for practical ambiguity functions. A receiver architecture consisting of a bandpass matched filter, square-law envelope detector, and specialized processing is proposed to produce images.

## Introduction

Inverse synthetic-aperture imaging in radar and sonar relies upon the relative motion between the transmitter, target, and receiver. In the usual approach, the target is illuminated by a series of transmitted pulses. The return for each pulse is a superposition of reflections from various locations on the target, with each location affecting the pulse by introducing a shift in delay and doppler. In this way, the cumulative return for each pulse is a complicated mixture of returns that is influenced by the shape and reflective properties of the target and its motion relative to the transmitter and receiver. The range and doppler histories of each of the reflected pulses are processed to produce a target's image.

The common approach is to use the same transmitted-pulse shape for each illumination of the target. Bernfeld [1] appears to be the *first to introduce the idea for radar imaging of modifying the pulse shape on* successive illuminations. He also suggested an approach for processing the reflected pulses so as to produce images of the target; his approach is based on an analogy he observed to the equations governing the data acquired in the x-ray tomographs currently being used for forming radiological images in medicine.

One deficiency in Bernfeld's novel approach is that, for the analogy to x-ray tomography to be accurate, the ambiguity function of the transmitted pulses must be highly concentrated along lines in the delay-doppler coordinates and, moreover, must have a constant amplitude along such lines. Thus, practical radar/sonar pulses having ambiguity functions with complicated sidelobe structures and a nonuniform amplitude along the principal lobe are not accommodated very well in the concept. The purpose of our paper is to note an alternative analogy to medical imaging where this restriction is relaxed. This extension to Bernfeld's idea may permit improved images to be formed for practical ambiguity functions.

## Model for Radar/Sonar and Imaging Problem

The model we use for radar/sonar imaging is the one described by Van Trees [2, Ch. 13]. The transmitted pulse has complex amplitude $E_t^{1/2}\tilde{f}(t)$, where $E_t$ is the transmitted energy. The reflected pulse at the receiver has complex amplitude $\tilde{s}(t)$, where

$$\tilde{s}(t) = \int_{-\infty}^{\infty} E_t^{1/2}\tilde{f}(t-\tau)\tilde{b}(t-\tau/2,\tau)\,d\tau,$$

where $\tilde{b}(t,\tau)$ is a zero-mean, complex-valued Gaussian process modeling a diffuse reflection interaction at the target. We will denote the power spectrum of $\tilde{b}(t,\tau)$ at range $\tau$ by $\sigma(f,\tau)$; this is the target's scattering function. Van Trees denotes this function as $S_{DR}(f,\tau)$. It is assumed that $\tilde{b}(t,\tau_1)$ and $\tilde{b}(t,\tau_2)$ are uncorrelated for $\tau_1 \neq \tau_2$. Thus, $\tilde{b}(t,\tau)$ models wide-sense stationary, uncorrelated scattering. The complex envelope of the received signal is $\tilde{r}(t) = \tilde{s}(t) + \tilde{w}(t)$, where $\tilde{w}(t)$ is a complex-valued, white Gaussian noise-process that is independent of $\tilde{b}(t,\tau)$ and which has spectral intensity $N_0$. For the work to be described here, we assume that $\tilde{r}(t)$ is first processed by a bandpass matched-filter square-law envelope detector (BPMF-SLED), in which case the average value of the result is given by [2, eqn (13.79)] $p(f,\tau) + N_0$, where

$$p(\tau,f) = E_t\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\sigma(\tau',f')\,a(\tau-\tau',f-f')\,d\tau'df' \tag{1}$$

[+]This work was supported by the Office of Naval Research under contract N00014-86-K-0370.

in which $(\tau, f)$ are the delay-doppler coordinates and $a(\tau, f)$ is the ambiguity function of the transmitted signal, which is the squared magnitude of the complex delay-doppler correlation function [2, eqn (10.18)]. In the absence of additive noise (i.e., $N_0 = 0$), the average output of this receiver processing is, therefore, the two-dimensional convolution of the target's scattering function with the ambiguity function of the transmitted signal.

Adopting Bernfeld's idea, we consider that the transmitted pulses which illuminate the target are chirp-rate modulated, by which we mean that each pulse has a particular linear FM chirp rate but the chirp rate is varied from pulse to pulse. The effect of changing the chirp rate of a pulse is to shear or tilt its ambiguity function in the delay-doppler coordinates [2, p. 290]. We shall denote this tilt by the parameter $\theta$, which depends on the chirp rate. We will assume in what follows that the pulse shape is changed along with the chirp rate in such a way that the only variation of the ambiguity function is to rotate it to an angle $\theta$ in the delay-doppler plane; it is unnecessary to maintain a fixed shape of the ambiguity function in general, but this assumption results in the closest analogy to emission tomography. In order to include this chirp-rate modulation in our notation, we shall now replace $\tilde{f}(t)$, $a(\tau, f)$, $p(\tau, f)$, and $\tilde{r}(t)$ by $\tilde{f}_\theta(t)$, $\tilde{a}_\theta(\tau, f)$, $\tilde{p}_\theta(\tau, f)$, and $\tilde{r}_\theta(t)$.

We may now state the delay-doppler imaging problem for radar and sonar as that of estimating the target's scattering function from the data $r_\theta(t)$ acquired for a series of target illuminations at $\theta_1$, $\theta_2$, ..., $\theta_N$. By temporarily introducing the simplifying assumptions that the additive noise is negligible and that the resulting data can be replaced by their average value, $p_\theta(\tau, f)$, the imaging problem becomes a deconvolution problem in which we are given the ambiguity function $a_\theta(\tau, f)$ and data $p_\theta(\tau, f)$ for several values of $\theta$, and we must solve (1) for the scattering function $\sigma(\tau, f)$. We have noted in [3] that this imaging problem is analogous to one encountered in positron-emission tomography.

## Model for Positron Emission Tomography and Imaging Problem

An equation having the same form as (1) occurs in positron-emission tomographic-imaging systems. In these systems, a positron-emitting radionuclide is introduced inside a patient, and the resulting activity is observed externally with an array of scintillation detectors that surround the patient in a planar, ring geometry. When a positron is produced in a radioactive decay, it annihilates with an electron producing two 511 kev photons that propagate at the velocity of light in opposite directions along a line. In systems under current development, both the line-of-flight and the differential time-of-flight of the annihilation photons are measured, which provides an estimate of the location of the annihilation. The result is that, in the absence of noise, the measurements are in the form of (1) with $\sigma(\tau, f)$ being the two-dimensional activity distribution to be imaged, and with $a_\theta(\tau, f)$ being the known error density associated with the measuring the location of an annihilation event [4]. For present systems, the error density is a two-dimensional elliptical-shaped Gaussian function with the long axis oriented at an angle $\theta$ corresponding to the line-of-flight of the annihilation photons, and data are collected for from 32 to 96 discrete angles over the range (0,180°). This density corresponds to the ambiguity function of a radar pulse having an envelope that is a Gaussian function and a phase that is a linear FM chirp.

In summary:

a. For delay-doppler radar/sonar imaging, we suppose that the target is illuminated by a sequence of radar pulses each having a distinct FM chirp rate corresponding to angles $\theta_1$, $\theta_2$, ..., $\theta_N$ spanning the range from 0 to 180° in the delay-doppler plane. A BPMF-SLED receiver produces $p_\theta(\tau, f)$. The ambiguity function $a_\theta(\tau, f)$ is known. The problem is to estimate the target's scattering function $\sigma(\tau, f)$ using the relationship (1).

b. For emission-tomography imaging when both time-of-flight and line-of-flight data are collected, we have as measurements $p_\theta(\tau, f)$ for angles $\theta_1$, $\theta_2$, ..., $\theta_N$ spanning 0 to 180°. The measurement-error density $a_\theta(\tau, f)$ is known. The problem is to estimate the activity distribution $\sigma(\tau, f)$ using the relationship in (1).

Computer algorithms for solving the imaging problem for positron-emission tomography with time-of-flight measurements have been under intensive development since about 1980, and they continue to evolve and be refined. The first such algorithm, called the ''confidence weighting algorithm'' [4] continues to be the one routinely used. Algorithms under current development are based on the joint maximization of likelihood and entropy [5,6]; these are solved iteratively and are, therefore, very demanding computationally, but the results appear to be superior. In the following section, we indicate how the confidence weighting algorithm would be applied for the radar/sonar imaging problem. A future publication will describe how the newer approaches might be used for radar imaging.

## Confidence-Weighted Algorithm

The confidence-weighted algorithm for processing $p_\theta(\tau, f)$ to estimate $\sigma(\tau, f)$ is as follows. There are two steps; since the algorithm is linear, these could be combined into one, but two are used in emission tomo-

graphy because of implementation considerations. The first step is to form a two-dimensional convolution of the data $p_\theta(\tau,f)$ obtained for each value of $\theta$ with a weighting function $w_\theta(\tau,f)$ and then to form the pixel-by-pixel sum of the results for each $\theta$; that is, we form the functions

$$f_\theta(\tau,f) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} p_\theta(\tau',f')w_\theta(\tau-\tau',f-f')d\tau'df',$$

and then we obtain the two-dimensional ''preimage'' $f(\tau,f)$ according to

$$f(\tau,f) = (1/\pi)\int_0^\pi f_\theta(\tau,f)d\theta.$$

The formation of this preimage generalizes the back-projection step of the ''(unfiltered) back-projection, post two-dimensional filtering'' approach to idealized line-integral (i.e., Radon transform) tomography; for ideal-line integral tomography the almost universally used approach, which yields he same final result, is to filter before back projecting, but this does not work well when line integrals are a poor approximation, as in time-of-flight tomography because of the dispersed point spread $p_\theta(\tau,f)$.

As described in [3] and [4], various weighting functions might be adopted, but for time-of-flight tomography, the choice used is $w_\theta(\tau,f) = a_\theta(\tau,f)$. For radar/sonar imaging, this corresponds to taking the value of the BPMF-SLED output $p_\theta(\tau,f)$ for a particular value of $(\tau,f)$ and distributing it over the delay-doppler plane according to the ambiguity function. This might be motivated by noting that if we interpret $\sigma(\tau,f)$ as the total reflectance power at $(\tau,f)$ and $a_\theta(\tau'-\tau,f'-f)$ as the fraction of reflectance power at $(\tau,f)$ appearing as measured power at $(\tau',f')$, then the quantity $p_\theta(\tau,f|\tau',f')$ defined by

$$p_\theta(\tau,f|\tau',f') = a_\theta(\tau'-\tau,f'-f)\sigma(\tau,f)\left[\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} a_\theta(\tau'-\tau,f'-f)\sigma((\tau,f)d\tau df\right]^{-1}$$

is the fraction of reflectance power at $(\tau,f)$ given the total measured power at $(\tau',f')$, and

$$\sigma(\tau,f) = (1/\pi)\int_0^\pi\left[\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} p_\theta(\tau,f|\tau',f')p_\theta(\tau',f')d\tau'df'\right]d\theta$$

is the total reflectance power at $(\tau,f)$ given the power measured at every $(\tau',f')$ and every value of $\theta$. Since a property of the ambiguity function is that it has unit volume [2, page 308], this expression is simply another way to write the trivial equality that $\sigma(\tau,f) = \sigma(\tau,f)$. However, if this expression is adopted as a basis for estimating an unknown scattering function $\sigma(\tau,f)$ given the measurements $p_\theta(\tau,f)$, and if we further presume that the scattering function embedded implicitly within the right-hand side of this expression is equally likely to be anything (i.e., uniform) before making any measurements, then this expression reduces to the preimage expression defined before.

The second processing step is to obtain the target's image from the preimage. The required operation corresponds to the filtering step of the ''back project then filter'' algorithm used to invert the Radon transform. The analogy to the confidence-weighting algorithm of positron-emission tomography suggests that this should be accomplished to within a resolution function $h(\tau,f)$, which defines a ''desired image'' according to

$$d(\tau,f) = \iint h(\tau-\tau',f-f')\sigma(\tau',f')d\tau'df'.$$

We have found that including such a resolution function is important in processing emission-tomography data as a way to trade off resolution and smoothing for noise suppression. In [4], a narrow, two-dimensionally, circularly symmetric Gaussian resolution filter is used. Let $d(\tau,f)$ denote the estimate of $d(\tau,f)$ obtained by processing the preimage $f(\tau,f)$. Also, let $D(u,v)$ and $F(u,v)$ denote the two-dimensional Fourier transforms of $d(\tau,f)$ and $f(\tau,f)$, respectively. Then, from [4, eqn. (17)],

$$D(u,v) = H(u,v)F(u,v)/G(u,v),$$

where $H(u,v)$ is the transform of $h(\tau,f)$ and $G(u,v)$ is the transform of the function $g(\tau,f)$ defined according to

$$g(\tau,f) = (1/\pi)\int_0^{\pi} a_\theta(\tau,f) w_\theta(\tau,f) d\theta.$$

The image $d(\tau,f)$ is obtained from $D(u,v)$ by a two-dimensional inverse Fourier transformation. The function $g(\tau,f)$ and $G(u,v)$ are precomputable since they depend only on the ambiguity function and the weighting function adopted to form the preimage and not on the measured data. For the choice $w_\theta(\tau,f) = a_\theta(\tau,f)$, the function $g(\tau,f)$ is the average over $\theta$ of the square of the ambiguity function. In [4], $a_\theta(\tau,f)$ is a two-dimensional asymmetric Gaussian function, and $g(\tau,f)$ is a Bessel function. The derivation in [4] does not require that $a_\theta(\tau,f)$ be Gaussian, but then $g(\tau,f)$ will usually need to be evaluated numerically for practical ambiguity functions.

## Processing Architecture

The architecture of the imaging algorithm we have described offers opportunities for parallelism and the use of specially designed processors for near real-time processing. Data acquired for each chirp rate $\theta$ can be processed in parallel and then summed to form the preimage $f(\tau,f)$. The convolutions required for each chirp rate can be implemented in systolic or other high speed architectures. The architecture is diagrammed in Fig. 1.

## Preliminary Results

We have implemented a preliminary computer simulation to begin testing our algorithm. Two targets have been simulated, a rotating rough disk and a rotating rough sphere. Only average quantities were included in this first simulation; variations due to both additive noise and statistical variations in the reflection process are presently being implemented. Processing was performed using both our new algorithm (CW) and the algorithm conventionally used for inverse synthetic-aperture radar (ISAR) imaging.

<u>disk</u>: The axis of rotation is normal to the disk and passes through its geometric center. The radar looks in the plane of the disk. The diameter of the disk is 6.4 meters. The rotation rate is selected to give a circular scattering function in the delay-doppler plane.

<u>sphere</u>: The axis of rotation is through its geometric center. The radar looks in the equatorial plane of the sphere. The diameter of the sphere is 6.4 meters. The rotation rate is selected to give a circular scattering function in the delay-doppler plane.

Figures 2(a) and 2(b) show the power scattering function for the rotating disk and its reconstruction using the CW algorithm, respectively. Each image is an array of 128 range by 128 cross-range resolution cells, with each cell being a square of dimension 0.1 meter on a side. The scattering function is contained in a square subarray of 64-by-64 pixels. A total of 128 distinct FM chirp rates were used such that the ambiguity function rotated at equal increments through 180°. The asymmetric Gaussian-shaped ambiguity function has dimensions of 28 pixels (FWHM) along the major axis and 4 pixels (FWHM) along the minor axis. The durations and chirp rates of the radar pulses were adjusted so that the ellipses describing their ambiguity functions had the same major and minor axis-dimensions at each rotation angle. The delay-doppler image obtained with the CW algorithm shows some degradation in resolution but no other major distortions are evident.

Figures 3(a) and 3(b) show the power scattering function for the rotating sphere and its reconstruction using the CW algorithm, respectively. Each image is an array of 128 range by 128 cross-range pixels, with each pixel being a square of dimension 0.1 meter on a side. The scattering function is contained in a square subarray of 64-by-64 pixels. A total of 128 distinct FM chirp rates were used such that the ambiguity function rotated through 180°. The durations and chirp rates of the radar pulses were adjusted so that the ellipses describing their ambiguity functions had the same major and minor axis-dimensions (24 by 4 pixels FWHM, respectively) at each rotation angle. The delay-doppler image obtained with the CW algorithm shows some degradation in resolution but no other major distortions are evident.

Figures 4(a), (b), and (c) show examples of $p_\theta(\tau,f)$ for $\theta$ = 0°, 45°, and 90° for the rotating sphere.

Figures 5(a) and 5(b) show the squared magnitude of the reflectivity (complex amplitude) function for the rotating disk and its reconstruction using the conventional ISAR algorithm. Each image is an array of 128 range by 128 cross-range resolution cells, with each cell being a square of dimension 0.1 meter on a side. The reflectivity function is contained in a square subarray of 32-by-32 pixels. A stepped-frequency radar-pulse sequence consisting of 128 square pulses, each of duration 85 ns separated in time and frequency by 3.4 ms and 11.7 MHz, respectively, is assumed. the center frequency is 3 GHz, and the target is illuminated by 128 such pulse sequences. For the simulation, we calculated the received signal from the target using discrete points to model the distributed reflectivity within resolution cells distributed evenly in a rectangular grid over the target. The received signal for each transmitted pulse is processed to generate the image by first sorting into range-resolution cells using a Fourier transform and then sorting the result into doppler resolution-cells by again using a Fourier transform on the data in each range cell. The resulting image shows some loss in resolution and an oscillatory artifact around the edge of the disk.

Figures 6(a) and 6(b) show the magnitude of the reflectivity function for the rotating sphere and its reconstruction using the conventional ISAR algorithm. Each image is an array of 128-by-128 resolution cells. The reflectivity function is contained in a square subarray of 32-by-32 pixels, with each pixel being a square of dimension 0.1 meter on a side. The same ISAR processing described for the disk was used. The resulting image shows loss in resolution and an oscillatory artifact around the edge of the disk.

We wish to emphasize that these results are our first ones, and they are very preliminary. We are encouraged by them because the images produced with the CW algorithm do seem to have less artifacts than those produced with the conventional ISAR algorithm. We expect that a greater difference in the resulting images will result when additive noise and statistical variations in the reflection process are included in our simulation, with the CW algorithm being superior because of the use of the matched filter for additive noise suppression and averaging over doppler rates in forming the image.

## References


1. N. Bernfeld, ''CHIRP Doppler Radar,'' Proc. IEEE, Vol. 72, No. 4, pp. 540-541, April 1984.

2. H. L. Van Trees, Detection, Estimation, and Modulation Theory, Part III,'' John Wiley and Sons, New York, 1971.

3. D. L. Snyder and H. J. Whitehouse, ''Delay-Doppler Radar Imaging Using Chirp-Rate Modulation,'' Proc. Dixieme Colloque Sur le Traitment du Signal et Ses Applications, Nice, France, May 1985.

4. D. L. Snyder, L. J. Thomas, Jr., and M. M. TerPogossian, ''A Mathematical Model for Positron Emission Tomography Systems Having Time-of-Flight Measurements,'' IEEE Trans. on Nuclear Science, Vol. NS-28, pp. 3575-3583, June 1981.

5. M. I. Miller and D. L. Snyder, ''The Applications of Maximum Entropy and Maximum Likelihood for the Solution of Incomplete and Noisy-Data Problems in Estimating Point-Process Intensities, Probability Densities, and Spectral Densities,'' IEEE Proceedings, in review.

6. M. I. Miller and D. L. Snyder, ''Iterative Algorithm and Architecture for Maximum Entropy and Its Relation to Maximum Likelihood, with Applications to Imaging Single Photon Radioactivity Distributions,'' Proc. SPIE, Advanced Algorithms and Architectures for Signal Processing, San Diego, Aug. 1986.

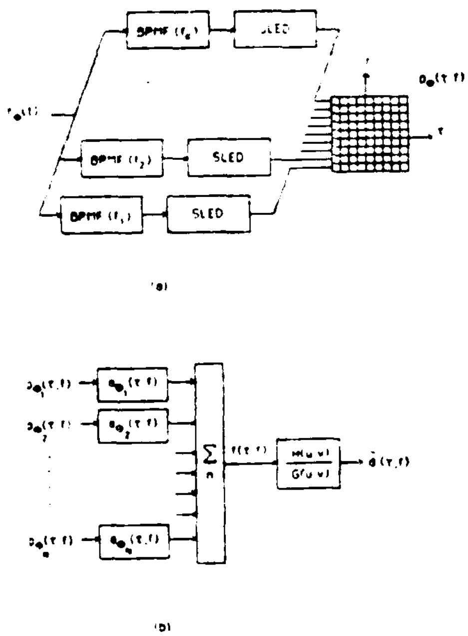




Figure 1. a) generate the delay-doppler function for each chirp rate. b) generate preimage by weighted back-projection and the desired estimate by post filtering.

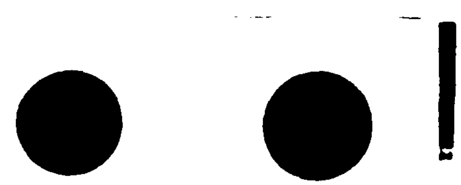


Figure 2. a) power scattering function for a rotating disk. b) reconstruction using the CW algorithm.

696 20

Figure 3. a) power scattering function for a rotating sphere. b) reconstruction using the CW algorithm.

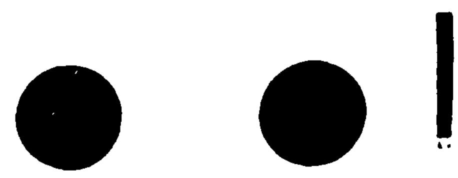


Figure 5. a) squared magnitude of the reflectivity function for a rotating disk. b) reconstruction using the standard ISAR algorithm.

Figure 4. BPMF-SLED delay-doppler output for a) 0°, b) 45°, and c) 90°.

Figure 6. a) squared magnitude of the reflectivity function for a rotating sphere. b) reconstruction using the standard ISAR algorithm.

## APPENDIX 2.

### ANNOTATED LISTINGS OF COMPUTER PROGRAMS DEVELOPED

APPENDIX 2A. PROGRAM LISTINGS FOR CONVENTIONAL ISAR SIMULATION

SIMULATION PERFORMED BY: Robert C. Lewis

A FORTRAN program was written to achieve the goal of producing conventional ISAR images in a laboratory simulation. The dimensions of simulated targets were specified and their scattering functions were supplied for use. The ISAR simulation contained two parts. The first inputs the scattering function and calculates the received signals from that target for all pulses and bursts transmitted. The second part processes the received signals using Fourier transform techniques, as described here, to produce an image data file. An image display routine is used to display the ISAR image on a color monitor.

An algorithm is presented in Figure A2.1 to describe the programming of the mixer output portion of the simulation. A two-dimensional array is used to store the resulting radar signals. Two analytic targets were chosen, a rough disk and a rough sphere. The field of view of the image was chosen to be 76.8 meters in both range and cross range dimensions. The resolution in both dimensions was chosen to be 60 cm. Therefore, the scattering function is a 128-by-128 array of reflected power values. The diameters of both the disk and sphere were chosen to be 38.4 m with both centered in the field of view. It was desired to write a computer program that would generate an image of size 256-by-256 pixels. This is required by the Fourier transform techniques used.

In this simulation, 128 pulses per burst and 128 bursts are used in the transmitted waveform. This covers the field of view given the resolution. The resulting two-dimensional array holding the received radar signal is of size 128-by-128. The range profiles are calculated by using a discrete Fourier transform (DFT) and the following property to get an inverse DFT:

```
Read in Scattering Function
Increment Burst Number m
|    K=1
|    Increment Resolution Cell  (r, f) = (i, j)
|    |    Increment Pulse Number n
|    |    |    Calculate Received Signal S(n)
|    |    |    S_k(n) = S_k-1(n) + S(n)
|    |    End Loop
|    |    K = K + 1
|    End Loop
|    Store S(n) in S(n, m)
End Loop
Normalize S(n, m) to max [S(n, m)] = 1.0
End
```

FIGURE A2.1 Algorithm Used for Received Signal Simulation

if $X(k)$ is the DFT of $x(n)$, then $Nx^*(k)$ is the DFT of $X^*(n)$, where ''*'' denotes complex conjugation. Also, in computing a range profile, the 128 point signal is placed in the center of a 256 point array, padded with zeros, for the inverse DFT operation. The resulting 256 point range profile, divided by the number of points, replaces the row of signal data in the two-dimensional array.

In the final step of calculations, the cross-range profiles of the image are made using a DFT on the columns of the two-dimensional array resulting from the last operation. However, a rearrangement of the data in the 256 point array is needed. Instead of placing the 128 data points in the center of the 256 point array, the first 64 data points are placed in the first 64 array locations, leaving the center array locations zeroed.

This is illustrated in Figure A2.2. After the DFT is computed, the zero

frequency location is in the first array location, so the data are rearranged with the last half of the 256 points interchanged with the first half. The magnitudes of the resulting complex numbers are stored in each column of the two-dimensional array. The resulting two-dimensional image data array is then used with a color scale to display the target's image.

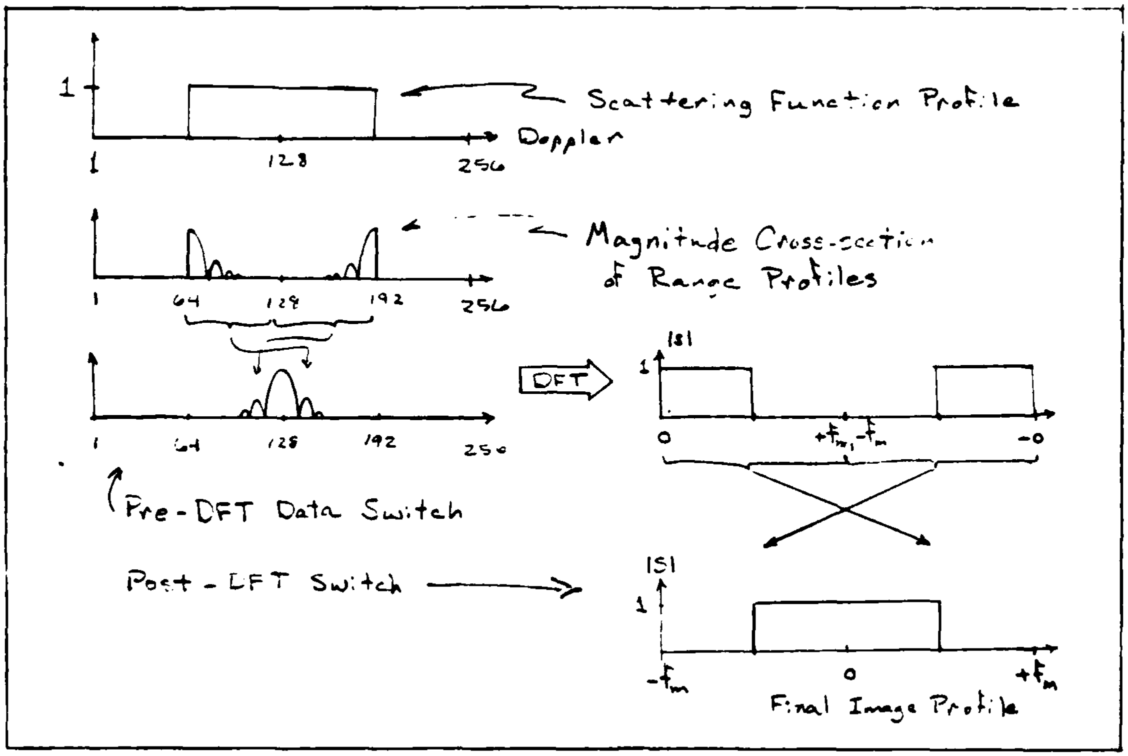



**FIGURE A2.2 Fourier Transform Techniques**

A listing of the computer program follows. In the main routine, PROC3, the target's scattering function is input and the radar received signal is calculated. This signal is normalized to peak magnitude in loops 110, 120,

130, and 140. Subroutine IMAGE2D is then called to reconstruct the image.

A brief description of each program and subroutine is as follows.

**PROC3:** calculates a mixer output signal from an input scattering function data file. A non-statistical target is assumed.

| | |
|---|---|
| **CRX, CRY** | location of center of rotation within the scattering function in cm. |
| **TT** | total time interval of all pulses and bursts |
| **BW** | bandwidth of a burst |
| **F0** | lowest frequency in a burst |
| **D** | step frequency increment |
| **N** | number of pulses in a burst, number of bursts |

**IMAGE2D:** performs ISAR processing to reconstruct an image from radar signals. The DFTs are implemented using the FFT algorithm.

**TARGET:** reads in the target's scattering function data.

```
      PROGRAM PROC7
      DIMENSION S(128),Q(128),SI(128)
      REAL LAMBDA,NA
      Complex FRAME
      Common /blk1/ LAMBDA(128,128)
      Common /blk2/ FRAME(256,256)

C**************************************************************
C*          STEPPED-FREQUENCY ISAR PROCESSING (WINDOWED)      *
C*               NON-STATISTICAL REFLECTIVITY                 *
C*                  BOB LEWIS, NOV. 1986                      *
C* input=16, output=17                                        *
C**************************************************************
      C=3.E8
      PI=3.1415927
      CRX=6.3
      CRY=6.3
      TT=9.23
      BW=0.25E9
      F0=2.875E9
      DF=BW/127.
      N=128

C************************************************************
C********* Read in target's scattering function.  ***********
C************************************************************
      CALL target

      L=65
      NA=REAL(N-1)
      DT=TT/128.
      T2=DT/127.
      T1=1./DF
      T12=T1/2.
```

```
C*****************************************************************
C****** Calculate mixer output signal from scattering funct of LAMBDA. **
C****** Loop 100 steps burst #, loops 70,80 step thru resolut of cells. **
C****** Loop 60 steps pulse #. MD signal is stored in FRAME.          **
C********************************************************************
      DO 100 T=0.,TT,DT
         DO 50 K=1,128
            S(K)=0.
            Q(K)=0.
            SI(K)=(K-1)*T2+T+T12
   50    CONTINUE
         DO 80 I=1,128
            X=(I-1)*.1-CRX
            DO 70 J=1,128
               IF(LAMBDA(I,J).EQ.0.)GO TO 70
               A=LAMBDA(I,J)**0.5
               Y=(J-1)*.1
               DELAY=6.6667E-9*Y
               DO 60 K=1,128
                  FI=(K-1)*DF+F0
                  VT2C=X/(12.8*DT*FI)
                  PI2FTAU=6.2832*FI*(DELAY-VT2C*(SI(K)+DELAY))
                  S(K)=S(K)+A*COS(PI2FTAU)
                  Q(K)=Q(K)+A*SIN(PI2FTAU)
   60          CONTINUE
   70       CONTINUE
   80    CONTINUE
         DO 90 K=1,128
            FRAME(L,K+64)=CMPLX(S(K),Q(K))
   90    CONTINUE
         L=L+1
  100 CONTINUE

C**************************************************************
C****** Normalize mixer output to peak magnitude = 1.0 *******
C**************************************************************
      PK=0.
      DO 120 L=1,256
         DO 110 K=1,256
            PK=MAX(PK,ABS(FRAME(L,K)))
  110    CONTINUE
  120 CONTINUE
      PKINV=1./PK
      DO 140 L=1,256
         DO 130 K=1,256
            FRAME(L,K)=FRAME(L,K)*PKINV
  130    CONTINUE
  140 CONTINUE

      CALL image2D
      stop
      END
```

```
      Subroutine  image2D
C*********************************************************************
C******** This subroutine reconstructs a radar  image from the    ****
C******** mixer output signal, by using Fourier transforms.       ****
C******** Windowing is used in this version.                      ****
C*********************************************************************
      Real image
      Complex f(256),frame
      Dimension W1(128)
      Common /blk2/ frame(256,256)
      Common /blk3/ image(256,256)

      M=8
      SF=5.961

      DO 10 I=1,128
         W1(I)=SF*EXP(-0.009*(I-64)**2)
 10   CONTINUE

      DO 120 i=1,256
         DO 100 j=1,256
            f(j)=conjg(frame(i,j))
100      CONTINUE
         DO 101 j=1,128

            f(j+64)=f(j+64)*W1(j)
101      CONTINUE
       CALL FFT(f,M)
         DO 110 j=1,256
            frame(i,j)=conjg(f(j))*0.00390
110      CONTINUE
120   CONTINUE

      DO 150 j=1,256
         DO 125 i=1,256
            f(i)=cmplx(0.,0.)
125      CONTINUE
          DO 130 i=1,64
             f(i+64)=frame(i+128,j)
             f(i+128)=frame(i+64,j)
130      CONTINUE
         DO 131 i=1,128
            f(i+64)=W1(i)*f(i+64)
131      CONTINUE
         CALL FFT(f,M)
         DO 140 i=1,128
            image(i+128,j)=(abs(f(i)))**2
            image(i,j)=(abs(f(i+128)))**2
140      CONTINUE
150   CONTINUE

      WRITE(17,1)((IMAGE(I,J),I=1,256),J=1,256

      Return
    1 FORMAT(1X,8E9.2)
      END
```

```
      SUBROUTINE TARGET
C*************************************************************
C******** READ IN TARGET'S SCATTERING FUNCTION.    ********
C*************************************************************
      REAL LAMBDA
      COMMON /BLK1/ LAMBDA(128,128)

      READ(16,1)((LAMBDA(I,J),I=1,128),J=1,128)

      RETURN
 1    FORMAT(1X,8E9.2)
      END
```

## APPENDIX 2B. PROGRAM LISTINGS FOR CONFIDENCE WEIGHTED ISAR SIMULATION

**SIMULATION PERFORMED BY: J. Trent Wohlschlaeger**

The software system developed to simulate the proposed processing algorithm consists of several separate programs, each designed to perform a specific step of the processing.

The programs are listed below in the order that they are run, along with a short comment on their function. For a more thorough desription of a particular program's use, please see the accompanying listings.

| Program | Description |
|---|---|
| 1. mkfilt | Generates ambiguity functions. Run once for a given set of parameters. |
| 2. spinball | Generates the scattering function for a rotating sphere. |
| 3a. detrtrn | Generates the p(theta)'s corresponding to radar returns from a deterministic reflection model. |
| OR | |
| 3b. rndmrtrn | Generates the p(theta)'s corresponding to radar returns from a random reflection model. |
| 4. cwasd2d | Performs the confidence weighted algorithm on the returns data generated in step 3. |
| 5. fnimg | Generates the final image by filtering the pre-image developed in step 4. |
| 6. vufnl | displays the final image. |

```
c*********************************************************************
c
c     Program Title        mkfilt.f
c
c     Written by           J. Trent Wohlschaleger
c     Date Written         March 21, 1985
c     Written for          Delay Doppler Radar Imaging
c
c     Program Intent
c               This program has several purposes.  The main
c        aim is to generate a lookup table that will be used
c        by another program to perform convolutions.  These
c        convolutions will be with a specific type of function,
c        namely a two-dimensional assymetric gaussian.  The
c        special nature of this function makes the task of
c        finding the look up table coefficients a little easier
c        (or faster) than for an arbitrary function.
c        While performing the convolution, only pixels
c        that fall within a certain range of the image pixel
c        currently being convolved will be updated.  This
c        range is chosen to be a certain number of standard
c        deviations, where a standard deviation is related
c        to the full width half max measurement error by
c        a constant.  The first step is therefore to find
c        which pixels fall within a given number of standard
c        deviations of a central point.  This is done by
c        generating an ellipse of isoprobability with the
c        appropriate semimajor and semiminor axes and testing
c        to see whether the center of a pixel lies inside
c        the ellipse.  If it does, the pixel's row and column
c        are tabulated.  This is accomplished for all "nang"
c        angles.
c               Then for all angles from the first (at angle
c        theta0) to the angle corresponding to theta0 + 45
c        degrees, for all of the pixel centers inside of
c        the ellipse and tabulated earlier, a two-dimensional
c        numerical integration is performed to find the volume
c        above the pixel under the gaussian function.  The
c        integral method is a simple riemann sum with "numint"
c        function evaluations in each direction for a total
c        of numint**2 function evaluations.  The volume
c        underneath the gaussian for a pixel becomes the
c        weighting coefficient or look up table coefficient for
c        that pixel.  Now, each pixel which is assigned a
c        coefficient must also have an address so that the
c        program which is performing the subsequent convolutions
c        will know where to apply this weight.  The address is
c        calculated by taking n (the image array size) times
c        the row in which the pixel is located plus the column.
c        Thus,
c
c               offset = (n*row) + col.
c
c        The columns and rows can be positive or negative, as
c        shown in the following example (for n = 5).
c
```

```
c        a(-2,-2)  a(-1,-2)  a(0,-2)  a(1,-2)  a(2,-2)
c
c        a(-2,-1)  a(-1,-1)  a(0,-1)  a(1,-1)  a(2,-1)
c
c        a(-2, 0)  a(-1, 0)  a(0, 0)  a(1, 0)  a(2, 0)
c
c        a(-2, 1)  a(-1, 1)  a(0, 1)  a(1, 1)  a(2, 1)
c
c        a(-2, 2)  a(-1, 2)  a(0, 2)  a(1, 2)  a(2, 2)
c
c        Note that the column index is shown before the row
c        index, and that the column number increases from
c        the left to the right in the image and the row number
c        increases from the top to the bottom.  The gaussian
c        function is assumed to be centered at a(0,0).  The
c        address (or offset) of pixel a(-1,-2) would be
c        (n*row) + col = (5*(-2)) + (-1) = -11.  The
c        addresses of the other pixels are defined similarly.
c        Note that earlier it was stated that the
c        numerical integration was performed only for the
c        gaussian function generated for angles from the
c        first (at angle theta0) to the angle corresponding
c        to theta0 plus 45 degrees.  This is not quite
c        true.  While the integration is performed for the
c        gaussian generated at only these angles, the integration
c        will produce results that have some symmetry.  The
c        volume over pixel a(1,1), for example, will be the
c        same as the volume over pixel a(-1,-1) for any given
c        angle.  This program takes advantage of this symmetry
c        and performs the integration only for roughly half of
c        the pixels falling inside of the ellipse.  Note that
c        the number of pixels inside of the ellipse will always
c        be odd.  Therefore ((numins(angle) + 1)/2) volumes
c        must be taken, where numins is the number of pixels
c        falling inside of the ellipse.
c                Fine.  Now we have the addresses and coefficients
c        for the angles from the first to the one corresponding
c        to the first plus 45 degrees.  Now we take advantage
c        of the fact that the next 45 degrees worth of coefficients
c        and addresses can be found merely by exchanging the roles
c        of the rows and columns found for the first batch of
c        angles.  Similarly, the last 90 degrees worth of addresses
c        and coefficients can be derived from the first 90 degrees'
c        results by merely changing the sign of the row.  All of
c        these symmetry properties are exploited to minimize the
c        run time.  Note that the bulk of the run time is
c        spent doing the numerical integration, but this need
c        be done only for approximately 1/8 th of the pixels
c        for which addresses and coefficients are calculated.
c                At the beginning of the program description,
c        I mentioned that the program has several purposes,
c        but I have mentioned only the generation of the
c        look up table itself.  Other functions performed
c        include indicating the size of the disk file the
c        table will require, indicating if any of the weighting
c        coefficients are negative (they should all be positive
```

```
c         ideally), and outputting the number of pixels included
c         inside the ellipse at each angle and the total volume
c         over these pixels.
c
c         Input Parameters:
c                 fwhme  = full width half max measurement error
c                 .         along the line of flight, in cm.
c                 fwhmb  = full width half max measurement error
c                           transverse to the line of flight, in cm.
c                 pix    = image pixel size, in cm.
c                 nostde = number of standard deviations worth of
c                           pixels for which addresses and coefficients
c                           are to be generated.
c                 nang = number of angles for which the table is
c                           to be generated.
c                 numint = number of intervals in which the gaussian
c                           function is to be subdivided for the
c                           numerical integration, in each direction.
c                 n      = image array size, i.e. the image is of
c                           size n x n.
c         The first four parameters are real floating point
c         numbers, and the last three are integer numbers.
c
c         input files
c         lu4 = parameter table (see above)
c
c         output files
c         lu3 = look up table containing addresses and coefficients
c                 of the two-dimensional gaussian for all "nang"
c                 angles
c
c         program structure
c         mkfilt, newfio
c*********************************************************************

        real coeff1(1000), coeff2(1000,128), confac, cosqth, costhe,
     *  del, fwhmb, fwhme, nostde, pi, pix, scale, semaj,
     *  semasq, semin, semisq. sigbsq, sigesq, sigmab, sigmae,
     *  sincos, sinthe, sisqth, theta, u, volpix, volume(128), x,
     *  xcent, xlolim, xmax, xmin, xrot, xrotsq, xsqfac, xsqmul,
     *  xyfac, xymul, y, ycent, ylolim, ymaxsq, yminsq, yrot,
     *  yrotsq, ysqfac, ysqmul
        integer*4 angnum, ang1, ang2, col, colmax, colmin,
     *   coltab(1000,128), index, iowrt, lu3, n,
     *   numins(128), numint, numneg, numrec, offset(1000),
     *   ranadd, row, rowmax, rowmin, rowtab(1000,128),
     *   totnum, xindx, yindx

        integer*4 outdes, nbytes, nb1
        integer*4 nang, nterms, totang, minor
        integer   ttout,ttin
        real      fwhme, fwhmb, cmatrx(-128:127,-128:127)

        include 'parm.com'

c set input/output constants and the value of pi.
```

```
      data lowrt/z'00000038'/
      data lu3/3/
      data raradd/0/
      data pi/3.14159265/
      data ttout/6/, ttin/5/

      open(unit=2,file='mkfilt.lst',status='unknown')

c Create output file for byte oriented I/O.
      call fcreat('Enter output file for filter coeffs',outdes)

      write(ttout, 99010)
      read (ttin,  99005) nang
      write(ttout, 99020)
      read (ttin,  99005) nterms
      write(ttout, 99030)
      read (ttin,  99006) fwhme
      write(ttout, 99040)
      read (ttin,  99006) fwhmb
      lowang = (nterms - 1)/2
      totang = nang*nterms

99005 format(i5)
99006 format(f20.10)
99010 format(' Enter number of angles ',$)
99020 format(' Enter number of terms to compute each angle ',$)
99030 format(' Enter FWHME (~ 6.0) ',$)
99040 format(' Enter FWHMB (~ 1.0) ',$)

      nostde = 3
      numint = 10
      n = imgdmt

c if the number of angles specified is too great or
c not evenly divisible by four, exit.  if the number of
c angles is too great but evenly divisible by four this
c program should be modified by changing the declared array
c sizes to fit the new requirements.  if the number of angles
c is not evenly divisible by four this program will not
c apply because of the extensive exploitation of symmetry
c properties described above.

         IF (nang .gt. 128) THEN
                write(ttout,15)
                GOTO 1075
         END IF
         IF (mod(nang, 4) .ne. 0) THEN
                write(ttout,20)
                GOTO 1075
         END IF
      nbytes = nang*4

c initialize the parameter indicating the number of
c contiguous records which will be required to store the
c lookup table.  then find the number of records which will
```

```
c be required due to saving the number of pixels inside the
c ellipse of isoprobability at each angle into the look up
c table.

          numrec = 0
          IF (mod((nang*4), 256) .eq. 0) THEN
            numrec = numrec + (nang*4/256)
         ELSE
            numrec = numrec + (nang*4/256) + 1
          END IF

c set the angles useful to exploiting the symmetry properties.

          ang1 = nang/4
          ang2 = nang/2

c find the standard deviations and their squares.

          confac = 2.0*sqrt(2.0*alog(2.0))
          sigmae = fwhme/confac
          sigmab = fwhmb/confac
          sigesq = sigmae*sigmae
          sigbsq = sigmab*sigmab

c find the increment for the function evaluation during
c the numerical integration and the scale factor to multiply
c the result by, which is partially due to the gaussian and
c partially due to the integration method.

          del = pix/float(numint)
          scale = (del*del)/(2.0*pi*sigmae*sigmab)

c find the semimajor and semiminor axes of the ellipse
c of isoprobability at the given number of standard deviations,
c to be used to delineate which columns and rows to search
c for pixels inside.

          semaj = nostde*sigmae
          semin = nostde*sigmab
          semasq = semaj*semaj
          semisq = semin*semin
          xmax = semaj
          xmin = -semaj
          colmax = int((xmax/pix) + 1.0)
          colmin = -colmax
          rowmax = colmax
          rowmin = -rowmax

c search for all of the pixels inside the ellipse at each
c angle and tabulate the row and column at which they occur.
c also, keep track of the total number for each angle.

        totnum = 0
        numrec = 0

        DO 1000 angnum = 1, (ang1 - 1)
```

```
          write(ttout, 22010)angnum
22010 format(' starting angle ',i5)

          DO 550, col = colmin, colmax
             DO 540, row = rowmin, rowmax
                cmatrx(row,col) = 0.0
540          CONTINUE
550       CONTINUE

          DO 990 minor = -lowang, lowang
            theta = (pi*float((angnum - 1)*nterms+minor)/float(totang))
     *           + (pi/2.0)
             costhe = cos(theta)
             sinthe = sin(theta)
             numins(angnum) = 0
             DO 1005 col = colmin, colmax
                xcent = pix*float(col)
                DO 1010 row = rowmin, rowmax
                   ycent = -pix*float(row)
                   xrot = (xcent*costhe) + (ycent*sinthe)
                   yrot = (-xcent*sinthe) + (ycent*costhe)
                   IF ((xrot .ge. xmin) .and. (xrot .le. xmax)) THEN
                      xrotsq = xrot*xrot
                      yrotsq = yrot*yrot
                      ymaxsq = semisq*(1.0 - (xrotsq/semasq))
                      yminsq = -ymaxsq
                      IF ((yrotsq .ge. yminsq)
     *              .and. (yrotsq .le. ymaxsq)) THEN
                         sisqth = sinthe*sinthe
                         sincos = sinthe*costhe
                         cosqth = costhe*costhe
                         xsqmul = -0.5*((sisqth/sigbsq)+(cosqth/sigesq))
                         xymul = ((sigesq-sigbsq)*sincos)/(sigesq*sigbsq)
                         ysqmul = -0.5*((sisqth/sigesq)+(cosqth/sigbsq))
                         u = 0.0
                         xlolim=pix*(float(col) - 0.5)
                         ylolim=-pix*(float(row) + 0.5)
                         DO 1025 xindx = 1, numint
                            x = xlolim + (del*(float(xindx) - 0.5))
                            xsqfac = x*x*xsqmul
                            DO 1030 yindx = 1, numint
                               y = ylolim + (del*(float(yindx)-0.5))
                               xyfac = x*y*xymul
                               ysqfac = y*y*ysqmul
                              u = u + exp(xsqfac + xyfac + ysqfac)
1030                        CONTINUE
1025                     CONTINUE
                         volpix = u*scale
                         cmatrx(row,col) = cmatrx(row,col) + volpix
                      END IF
                   END IF
1010            CONTINUE
1005         CONTINUE
990       CONTINUE

           [illegible] = 0
```

```
            volume(angnum) = 0.0
            DO 990, col = colmin, colmax
               DO 880, row = rowmin, rowmax
                  IF (cmatrx(row,col) .ne. 0.0) THEN
                     index = index + 1
                     rowtab(index, angnum) = row
                     coltab(index, angnum) = col
                     coeff2(index, angnum) = cmatrx(row,col)/float(nterms)
                     volume(angnum) = volume(angnum) + coeff2(index,angnum)
                  END IF
880            CONTINUE
990         CONTINUE
            numins(angnum) = index

            IF (numins(angnum) .gt. 1000) THEN
               write(ttout,25)
               GOTO 1075
            END IF

c Use symmetry to turn coefficients around a bit.

            DO 1035 index = (((numins(angnum) + 1)/2) + 1), numins(angnum)
               coeff2(index, angnum) =
     *            coeff2((index - ((numins(angnum) + 1)/2)), angnum)
               rowtab(index, angnum) = -rowtab((index -
     *            ((numins(angnum) + 1)/2)), angnum)
               coltab(index, angnum) = -coltab((index -
     *            ((numins(angnum) + 1)/2)), angnum)
1035        CONTINUE
            totnum = totnum + numins(angnum)
            IF (mod((numins(angnum)*4), 256) .eq. 0) THEN
               numrec = numrec + (2*(numins(angnum)*4/256))
            ELSE
               numrec = numrec + (2*((numins(angnum)*4/256) + 1))
            END IF

            write(ttout,45) angnum, numins(angnum), volume(angnum)
1000  CONTINUE

c calculate the number of contiguous records required to
c store the entire look up table.

         write(ttout,30) numrec

c output a report indicating all of the input parameters
c selected.

         write(ttout,35) fwhme, fwhmb, pix, nostde, nang, numint, n
         write(ttout,40)


c find the coefficients, rows, and columns for the second
c 45 degrees worth of angles using the symmetry arguments
c stated earlier.

         DO 1040 angnum = (ang1 + 2), (ang2 + 1)
```

```
            numins(angnum) = numins(ang2 + 2 - angnum)
            volume(angnum) = volume(ang2 + 2 - angnum)
               DO 1045 index = 1, numins(angnum)
                  coeff2(index, angnum) = coeff2(index,
     *                (ang2 + 2 - angnum))
                  rowtab(index, angnum) = coltab(index,
     *                (ang2 + 2 - angnum))
                  coltab(index, angnum) = rowtab(index,
     *                (ang2 + 2 - angnum))
1045           CONTINUE
               write(ttout,45) angnum, numins(angnum), volume(angnum)
1040      CONTINUE

c find the coefficients, rows, and columns for the
c last 90 degrees worth of angles using the symmetry arguments
c stated earlier.

         DO 1050 angnum = (ang2 + 2), nang
            volume(angnum) = volume(nang + 2 - angnum)
            numins(angnum) = numins(nang + 2 - angnum)
            DO 1055 index = 1, numins(angnum)
               coeff2(index, angnum) = coeff2(index,
     *           (nang + 2 - angnum))
               rowtab(index, angnum) = -rowtab(index,
     *           (nang + 2 - angnum))
               coltab(index, angnum) = coltab(index,
     *           (nang + 2 - angnum))
1055        CONTINUE
            write(ttout,45) angnum, numins(angnum), volume(angnum)
1050     CONTINUE
         write(ttout,50) totnum
         write(ttout,55) numneg

c save the array containing the number of pixels inside
c the ellipse at each angle into the look up table.  this will
c be used to minimize the number of evaluations that must be
c performed in the subsequent program implementing the
c convolution.  note that the number of updates is not the
c same for every angle.
c
         call fwrite(outdes, numins, nbytes)

c transfer the coefficients into a one-dimensional array
c to facilitate transfer to disk, and calculate the address
c or offset.  then save on disk.


         DO 1060 angnum = 1, nang
            write(ttout,9022)angnum
            DO 1065 index = 1, numins(angnum)
               coeff1(index) = coeff2(index, angnum)
               offset(index)=(n*rowtab(index,angnum))+coltab(index,angnum)
9021     format(' ind =',i6' n =',i6' rowtab =',i6,' coltab =',i6
        *,       ' offset =', 10)
9022     format(' making indices, starting angle ',i5)
1065        CONTINUE
```

```
          nb1 = numins(angnum)*4
          call fwrite(outdes, coeff1, nb1)
          call fwrite(outdes, offset, nb1)
1060  CONTINUE

c       record a permanent record of the look up table generation
c       on the printer.

        write(2,35) fwhme, fwhmb, pix, nostde, nang, numint, n
        write(2,30) numrec
        write(2,40)
        DO 1070 angnum = 1, nang
          write(2,45) angnum, numins(angnum), volume(angnum)
1070  CONTINUE
        write(2,50) totnum
        write(2,55) numneg

c terminate.

1075  CONTINUE
        close(unit=2)
        call fclose(luout)
        stop

5       format(f)
10      format(i)
15      format(/,
     * '  number of angles too high for available array space',
     * '  program terminated.',/)
20      format(/,
     * '  nang must be evenly divisible by four. program aborted.',/)
25      format(/,
     * '  the number of pixels within kernel of gaussian error density',/,
     * '  exceeds array size. program aborted.',/)
30      format(/,
     * '  look-up table requires',i4,' contiguous records.')
35      format(/,
     * '  longitudinal fwhm = ',f7.3,'cm',/,
     * '  transverse fwhm = ',f7.3,'cm',/,
     * '  pixel size = ',f6.3,'cm',/,
     * ' ',f4.1,' standard deviations of error density included',
     * ' in kernel.',/,
     * '  look-up table generated for data taken from ',i3,' angles.',/,
     * '  riemann sum uses ',i3,' subdivisions per pixel.',/,
     * '  dimension of image array assumed to be ',i3,/)
40      format(/,
     *    '  angle      number of pixels     volume over          ',/,
     *    '  number      inside ellipse      these pixels         ',/)
45      format(3x, i3, 13x, i4, 13x, g12.5)
50      format(17x, i6)
55      format(/,i3,' negative coefficients were found.',/)

        end
```

```
C*******************************************************************
C
C     PROGRAM TITLE      spinball.f
C
C     WRITTEN BY         J. Trent Wohlschlaeger
C     DATE WRITTEN       3/12/86
C     WRITTEN FOR        Delay Doppler Radar Imaging
C
C
C
C     PROGRAM INTENT
C              THIS PROGRAM CREATES A N X N REAL ARRAY
C       REPRESENTING A SPINNING BALL.  THE ARRAY IS THEN STORED ON DISK.
C       PIXELS ARE DIVIDED INTO 100 (10X10) SUBPIXELS.
C       IF THE CENTER OF A SUBPIXEL IS INCLUDED IN THE REGION, THEN
C       THE PIXEL VALUE IS INCREMENTED BY 1.  THUS EACH PIXEL HAS A
C       VALUE RANGING FROM 0 TO 100.
C
C       REGION   DESCRIPTION      GEOMETRY
C       ------   -----------      --------
C
C     REGION 1     SURFACE OF     CIRCLE
C                  SPINNING BALL  CENTER = (0,0)
C                                 RADIUS = 8.00 CM.
C
C       INPUT FILES
C       LU2 = UNIT TO RECEIVE ANSWERS FROM
C
C       OUTPUT FILES
C       LUN = IMAGE FILE TO CONTAIN "SPINNING-BALL" TARGET
C
C*******************************************************************

      common  /files/ lun
      Implicit Logical (A-Z)
      REAL LAMBDA(65536)
      INTEGER N
      integer lun
      data    lun /-1/

      call fcreat('Enter output file for target model (spin_ball.???)',
     *       lun)

      PRINT *, '  PLEASE INPUT THE IMAGE ARRAY SIZE. (256) '
      READ  *, N
      PRINT *, '  THE IMAGE IS ', N, ' BY ', N, ' PIXELS.'

      CALL bldball (lambda, n)

      END


      Subroutine bldball (lambda, n)
C        Call actual mainline as a Subroutine to take advantage of
C        Fortran's adjustable array dimensions for lambda

      common      /files/ lun
```

```
      Implicit Logical (A-Z)
      REAL HAFNP1, LAMBDA(n,n), R1, RADSQ, R1SQ, SUM, X, XSQ, Y, YSQ, PI
      INTEGER I, J, N, NBYTES, i2, j2, N2
      integer lun

      Parameter (PI = 3.14159265)
      DATA R1   /32.0/
      DATA R1SQ /1024.0/

      HAFNP1 = 0.5 * FLOAT(N + 1)

      DO 11 I = 1, N
        DO 15 J = 1, N
           LAMBDA(I,J) = 0.0
15       CONTINUE
11    CONTINUE

      N2 = N/2
      DO 1000 I = 1, N2
         PRINT *, ' Starting Column ', i
        DO 1010 I2 = 0, 9
           X = (I + i2/10.0 - HAFNP1)
           XSQ = X**2
           DO 1020 J = 1, N
              DO 1030 J2 = 0, 9
                 Y = -(J + j2/10.0 - HAFNP1)
                 YSQ = Y**2
                 RADSQ = XSQ + YSQ
                 IF (RADSQ .LT. R1SQ) THEN
C    PIXEL CENTER LIES WITHIN REGION 1.
                    LAMBDA(I,J) = LAMBDA(I,J) +
     *                 (1.1 - (XSQ + YSQ)/R1SQ)**-0.5*abs(X/R1)
                 END IF
1030          CONTINUE
1020       CONTINUE
1010     CONTINUE
1000  CONTINUE

C      SET THE TOTAL INTENSITY.

      SUM = 0.0
      DO 2020 I = 1, N
        DO 2030 J = 1, N
           SUM = SUM + LAMBDA(I,J)
2030     CONTINUE
2020  CONTINUE
      PRINT *, ' Total Intensity generated:  ', sum

C      SAVE THE ARRAY LAMBDA, WHICH IS FILLED WITH THE INTENSITY
C      LAMBDA OF EACH PIXEL FOR THE REFLECTION PROCESS, ON THE
C      DISK.

      call frew(lun)
      nbytes = n*n*4
      call fwrite(lun, lambda, nbytes)
```

```
      Return
      END
```

```
c*****************************************************************
c
c     Program title        cwddrreturn.f
c
c     Written by           J. Trent Wohlschlaeger
c     Date written         May 23, 1986
c     Written for          DDR
c
c     Program intent
c               This program convolves a set of ambiguity functions
c          with a scattering function to produce a set of deterministic
c          Delay-Doppler radar returns, as would be obtained by passing a
c          radar signal through a bank of BPMF-SLED's.
c
c     Input files
c          lutab = look up table containing the coefficients and
c                  addresses used to implement the convolution
c                  of a scatter function  array with the two-dimensional
c                  ambiguity function.  The table has two parts.
c                  The first part is an Integer array with
c                  namb elements, where namb is the total
c                  number of ambiguity functions.  This array indicates
c                  how many addresses and coefficients are
c                  present in the second part of the table for
c                  each ambiguity function.  The second part of the table
c                  consists of the coefficients followed by
c                  the addresses, for each ambiguity function.
c                  The table generator is mkfilt.f.
c
c*****************************************************************
c
      Implicit Logical (A-Z)
      Include 'parm.com'
      Common /files/ luns, lutab

      Character time*24

      Real scatter(rtnnsiz), returns(imgdmt**2), coeff(5000), pct
      Integer address, ambnum, colend, colst, i,
     +        lutab, namb, luns (10), lutim,
     +        nsq, numins(128), offset(5000),
     +        pixend, pixfir, pixnum, pixst,
     +        row, rowend, rowst,
     +        width
      Integer nbytes, ranadd, nzero, iord
      Integer ttout,ttin
      Integer length,lceof, strtamb, rtncnt

      Data IORD   /Z'00000058'/
      Data ttout /6/, ttin /5/
      Data luns  /10*-1/

      lutim = 1
      CALL fcreat('Enter output header file name        ',luns(8))
      CALL fcreat('Enter output Data file name          ',luns(3))
      CALL fopen ('Enter file containing target model   ',luns(4))
```

```
      CALL fopenr('Enter file containing filters         ',lutab)
      PRINT *, ' Enter number of Returns '
      read (ttin,  2010) namb
2010  Format(i5)

      strtamb = 1

c      Read in the first part of the lookup table, which
c      indicates the number of coefficients and addresses stored
c      in the second part of the table, for each ambiguity function.

      length = 4*namb
      CALL fread(lutab, numins, length, lceof, rtncnt)
c
      open(unit=lutim,file='detrtrn.tim',status='unknown')
      rewind lutim
      CALL fdate(time)
      ambnum = 1
      write(lutim,9007) ambnum,time(1:24)
      close(lutim)

      colst = 1 + (imgdmt - imgdim)/2
      colend = imgdmt - (imgdmt - imgdim)/2
      rowst = colst
      rowend = colend

      nsq = imgdmt*imgdmt
      width = colend - colst
      pixfir = (imgdmt*(rowst - 1)) + colst

c      Main loop.  The cw algorithm for each Return is performed
c      here.

      NBYTES = nsq*4
      ranadd = 0
      CALL SYSIO (PBLK, IORD, luns(4), scatter, NBYTES, RANADD)

      DO 1005 ambnum = strtamb, namb
         write(ttout,15) ambnum, namb, time(12:19)
        DO 1000 pixnum = 1, imgdmt**2
           returns(pixnum) = 0.0
1000     CONTINUE
         length = numins(ambnum)*4
         CALL fread(lutab, coeff,  length, lceof, rtncnt)
         CALL fread(lutab, offset, length, lceof, rtncnt)
         write(ttout,9900) numins(ambnum)
9900  Format(' non-zero filter pixels for this Return: ',i5)
         nzero = 0
         pixst = pixfir
         DO 1015 row = rowst, rowend
            pixend = pixst + width
            DO 1020 pixnum = pixst, pixend
               IF (scatter(pixnum) .gt. 0.0) THEN
                  nzero = nzero + 1
                  DO 1025 i = 1, numins(ambnum)
                     addres = pixnum + offset(i)
```

```
                         returns(address) = returns(address)
     *                                  + (scatter(pixnum)*coeff(i))
1025                     CONTINUE
                    END IF
1020              CONTINUE
                  pixst = pixst + imgdmt
1015          CONTINUE
              pct = 100.*float(nzero)/float(imgdim**2)
              write(ttout,9038) pct,imgdim,imgdim
9038      Format(1x,f7.2,'% of pixels in the ',i3,' by ',i3,
     *                    ' image were nonzero.')
              CALL fdate(time)
              open(unit=lutim,file='detrtrn.tim',status='unknown')
              write(lutim,9007) ambnum,time(1:24)
9007      Format(' Begin processing Return #',i2,' at ',a)
              close(unit=lutim)
             CALL output(returns)
1005      CONTINUE

          CALL fclose(lutab)
          stop

15        Format(/,
     *          '  Return Construction algorithm.      ',/,
     *          '     Return number:  ',i2,' out of ',i3,'.' ,/,
     *          '     Start time:     ',a)

          END
```

```
c*****************************************************************
c
c     Program title        rndmrtrn.f
c
c     Written by           J. Trent Wohlschlaeger
c     Date written         Nov 5, 1986
c     Written for          Delay_doppler Radar Imaging group
c
c     Program intent
c             This program convolves a set of ambiguity functions
c         with a scattering function to produce a set of random
c         Delay-Doppler radar returns, as would be obtained by passing a
c         radar signal through a bank of BPMF-SLED's.
c
c     Input files
c         lutab = look up table containing the coefficients and
c                 addresses used to implement the convolution
c                 of an scatter function  array with the two-dimensional
c                 ambiguity function.  The table has two parts.
c                 The first part is an Integer array with
c                 namb elements, where namb is the total
c                 number of ambiguity functions.  This array indicates
c                 how many addresses and coefficients are
c                 present in the second part of the table for
c                 each ambiguity function.  The second part of the table
c                 consists of the coefficients followed by
c                 the addresses, for each ambiguity function.
c                 The table generator is mkfilt.f.
c
c     program structure
c         rndmrtrn
c
c******************************************************************
c
      Implicit Logical (A-Z)
      Include 'parm.com'
      Common /files/ luns, lutab

      Character time*24

      Real scatter(rtrnsiz), returns(imgdmt**2), coeff(5000), pct
      Complex creturn(imgdmt**2), b, gauss
      Integer irand1, irand2
      Integer addres, ambnum, colend, colst, i,
     *        lutab, namb, luns (10), lutim,
     *        nsq, numins(128), offset(5000),
     *        pixend, pixfir, pixnum, pixst,
     *        row, rowend, rowst,
     *        width
      Integer nbytes, ranadd, nzero, iord
      Integer ttout,ttin,
      Integer length,lceof, strtamb, rtncnt

      Data IORD   /x'00000058'/
      Data ttout  /6/, ttin /5/
      Data luns   /10*-1/
```

```
      Data irand1 /322491219/
      Data Irand2 /1/

      lutim = 1
      CALL fcreat('Enter output header file name         ',luns(8))
      CALL fcreat('Enter output Data file name           ',luns(3))
      CALL fopen ('Enter file containing target model    ',luns(4))
      CALL fopenr('Enter file containing filters         ',lutab)
      PRINT *, ' Enter number of ambiguity functions '
      read (ttin,  2010) namb
 2010 Format(i5)

      strtamb = 1

c      Read in the first part of the lookup table, which
c      indicates the number of coefficients and addresses stored
c      in the second part of the table, for each ambiguity function.

      length = 4*namb
      CALL fread(lutab, numins, length, lceof, rtncnt)
c
      open(unit=lutim,file='rndmtime.dat',status='unknown')
      rewind lutim
      CALL fdate(time)
      ambnum = 1
      write(lutim,9007) ambnum,time(1:24)
      close(lutim)

      colst = 1 + (imgdmt - imgdim)/2
      colend = imgdmt - (imgdmt - imgdim)/2
      rowst = colst
      rowend = colend

      nsq = imgdmt*imgdmt
      width = colend - colst
      pixfir = (imgdmt*(rowst - 1)) + colst

c      Main loop.  The convolution algorithm for each ambiguity function
c      is performed here.

      NBYTES = nsq*4
      ranadd = 0
      CALL SYSIO (PBLK, IORD, luns(4), scatter, NBYTES, RANADD)

      DO 1005 ambnum = strtamb, namb
         write(ttout,15) ambnum, namb, time(12:19)
        DO 1000 pixnum = 1, imgdmt**2
           creturn(pixnum) = 0.0
 1000    CONTINUE
         length = numins(ambnum)*4
         CALL fread(lutab, coeff,  length, lceof, rtncnt)
         CALL fread(lutab, offset, length, lceof, rtncnt)
         write(ttout,9900) numins(ambnum)
 9900 Format(' non-zero filter pixels for this Return: ',i5)
         nzero = 0
         pixst = pixfir
```

```
          DO 1015 row = rowst, rowend
             pixend = pixst + width
             DO 1020 pixnum = pixst, pixend
                IF (scatter(pixnum) .gt. 0.0) THEN
                   nzero = nzero + 1
                   CALL nrmran (irand1, irand2, gauss)
                  b = scatter(pixnum)*gauss
                  DO 1025 i = 1, numins(ambnum)
                      addres = pixnum + offset(i)
                      creturn(addres) = creturn(addres)
     *                                 + (b*coeff(i))
1025               CONTINUE
               END IF
1020         CONTINUE
             pixst = pixst + imgdmt
1015      CONTINUE
          pct = 100.*float(nzero)/float(imgdim**2)
          write(ttout,9038) pct,imgdim,imgdim
9038  Format(1x,f7.2,'% of pixels in the ',i3,' by ',i3,
     *             ' image were nonzero.')
          CALL fdate(time)
          open(unit=lutim,file='rndmtime.dat',status='unknown')
          write(lutim,9007) ambnum,time(1:24)
9007  Format(' Begin processing Return #',i2,' at ',a)
          close(unit=lutim)
         DO 3000 pixnum = 1, imgdmt**2
           returns (pixnum) = abs (creturn(pixnum))
3000      CONTINUE
         CALL output(returns)
1005  CONTINUE

      CALL fclose(lutab)
      stop

15    Format(/,
     *         '    Return Construction algorithm.     ',/,
     *         '       Return number:  ',i2,' out of ',i3,'.' ,/,
     *         '       Start time:     ',a)

      END
```

```
c**************************************************************************
c
c     Program title        cwasd2d.ftn
c
c     Written by           J. Trent Wohlschlaeger
c     Date written         Nov. 5, 1986
c     Written for          Delay-Doppler Radar Imaging
c
c     Program intent
c               This program applies the confidence-weighting
c          algorithm to the reconstruction process in Delay-Doppler
c          radar imaging to produce an estimate of the desired image.
c               The "returns" data, sorted by burst, and converted into
c          ordinary (x,y) coordinates, must be available.
c
c     Input files
c          ludat = returns data in squeezed format
c          lutab = look up table containing the coefficients and
c                  addresses used to implement the convolution
c                  of an image array with the two-dimensional
c                  ambiguity function.  The table has two parts.
c                  The first part is an integer*4 array with
c                  nburst elements, where nburst is the total
c                  number of bursts.  This array indicates
c                  how many addresses and coefficients are
c                  present in the second part of the table for
c                  each burst.  The second part of the table
c                  consists of the coefficients followed by
c                  the addresses, for each burst.
c                  The table generator is mkfilt.f.
c
c     program structure
c          cwasd2d
c            genrtrn
c
c**************************************************************************
c
      common /rtrnarea/  rtrnpix, noadd, noval
      include 'parm.com'

      character time*24

      integer rtrnpix(128), noadd(10000)
      Real noval(10000)
      real rtrn(rtrnsiz), cwpimg(imgdmt**2), coeff(9000)

      integer addres, brstnum, colend, colst, i, lutab, nburst,
     *   nsq, numins(128), offset(9000), pixend, pixfir, pixnum, pixst,
     *   rowend, rowst, width
      integer rtrnhead,rtrndata,ttout,ttin
      integer*2 header(128)
      integer ludat,lutab,luimg,length,imgflg,lceof, strtbrst, rtncnt

      data      ttout /6/, ttin /5/
c
      lutim = 1
```

```
      CALL fopenr("Enter file with filters ",lutab)
      CALL fcreat("Enter output file for confidence weighted preimage ",
     *           luimg)
      CALL fopenr("Enter returns header file ",rtrnhead)
      CALL fopenr("Enter returns data file   ",rtrndata)
      write(ttout, 2000)
      read (ttin,  2010) nburst
2000      format(' Enter number of bursts ',$)
2010      format(i5)

c
      do 1000 pixnum = 1, imgdmt**2
         cwpimg(pixnum) = 0.0
1000      continue

      strtbrst = 1

c      Read in the first part of the lookup table, which
c      indicates the number of coefficients and addresses stored
c      in the second part of the table, for each burst.
c      Do the same for the returns data.

      length = 4*nburst
      CALL fread(lutab,  numins, length, lceof, rtncnt)
      CALL fread(rtrnhead, rtrnpix,  length, lceof, rtncnt)
c
      open(unit=lutim,file='cwtime.dat',status='unknown')
      rewind lutim
      CALL fdate(time)
      brstnum = 1
      write(lutim,9006) brstnum,time(1:24)
9006      format(' Begin processing burst #',i3,' at ',a)
      close(lutim)

      colst = 1 + (imgdmt - imgdim)/2
      colend = imgdmt - (imgdmt - imgdim)/2
      rowst = colst
      rowend = colend
c
      nsq =imgdmt*imgdmt
      width = colend - colst
      pixfir = (imgdmt*(rowst - 1)) + colst

c      Main loop.  The cw algorithm for each burst is performed
c      here.

      do 1005 brstnum = strtbrst, nburst
         write(ttout,15) brstnum, nburst, time(12:19)
         CALL genrtrn(brstnum, rtrndata, rtrn)
         write(ttout,9900)numins(brstnum)
9900      format(' non-zero filter pixels for this burst: ',i5)
         length = numins(brstnum) * 4
         CALL fread(lutab, coeff,  length, lceof, rtncnt)
         CALL fread(lutab, offset, length, lceof, rtncnt)

         nzero = 0
```

```
            pixst = pixfir
            do 1015 row = rowst, rowend
                  pixend = pixst + width
                  do 1020 pixnum = pixst, pixend
                        if (rtrn(pixnum) .gt. 0.0) then
                            nzero = nzero + 1
                            do 1025 i = 1, numins(brstnum)
                              addres = pixnum + offset(i)
                              cwpimg(addres) = cwpimg(addres)
     *                              + (rtrn(pixnum) * coeff(i))
1025                            continue

                        end if
1020                  continue
                  pixst = pixst + imgdmt
1015            continue
            pct = 100.*float(nzero)/float(imgdim**2)
            write(ttout,9038) pct,imgdim,imgdim
9038      format(1x,f7.2,'% of pixels in the ',i3,' by ',i3,
     *          ' image were nonzero.')

c     update the cumulative sum for the number of bursts so far.
c     In case of system crash, we can re-start in the middle.
            CALL frew(luimg)
            length = 256
            CALL fwrite(luimg,header,length)
            length = 4*imgdmt**2
            CALL fwrite(luimg,cwpimg,length)

            CALL fdate(time)
            open(unit=lutim,file='cwtime.dat',status='unknown')
            write(lutim,9007) brstnum,time(1:24)
9007      format(' Finished processing burst #',i3,' at ',a)
            close(unit=lutim)
1005      continue

c     save the confidence-weighted preimage onto disk.
c
       imgflg = -1
       length = 256
       CALL frew(luimg)
       CALL fwrite(luimg,header,length)
       length = 4*imgdmt**2
       CALL fwrite(luimg,cwpimg,length)
c
       CALL fclose(luimg)
       CALL fclose(lutab)
       CALL fclose(ludat)
c

       stop
```



```
15       format(/,
     *   '  CW reconstruction algorithm.                          ',/,
     *   '     burst number:                  ',i3,' out of ',i3,'.' ,/,
```

```
     *   '      Start time:                     ',a)
      end
```

```
c**********************************************************************
c
c     Program title          finimg.f
c
c     Written by             J. Trent Wohlschlaeger
c     Date written           Nov. 16, 1986
c     Written for            Delay-Doppler Radar Group
c
c     Program intent
c                 this program performs the filtering for the
c           reconstruction from a preimage to a final image.
c
c           type = 0 ==>  confidence-weighted filtering.
c           type = 1 ==>  most-likely-position filtering.
c           type = 2 ==>  convolve with a circularly symmetric
c                         gaussian to form the desired image.
c
c           twodim = 0 ==>  do not form a two-dimensional cw
c                           preimage by an extra convolution
c                           with a circularly symmetric
c                           gaussian.
c           twodim = 1 ==>  form a two-dimensional cw preimage
c                           and filter.
c
c           The equation used for reconstruction is
c
c                          d = f*h/g
c
c           where
c           d is the fourier transform of the desired image,
c           f is the fourier transform of the preimage,
c           h is the fourier transform of a resolution cell, and
c           g is the fourier transform of the filter.
c                 This program reads in f (in the space domain),
c           takes the two-dimensional fourier transform, generates
c           the filter h/g, multiplies f by h/g, and then
c           inverse fourier transforms the result to form the
c           final image, which is then stored on disc.
c           If type = 2, then g is set to 1, so the effect is
c           to convolve the preimage (which should be the
c           scattering function in this case) with a
c           circularly symmetric gaussian, to form the desired
c           image.
c
c     Program structure
c           finimg
c              fft
c              expbes
c              besj1
c
c**********************************************************************

      Implicit Logical (A-Z)
      include 'parm.com'

      Real armag(imgdmt,imgdmt), arreal(imgdmt,imgdmt), besj1,
```

```
     *          cwfilt, expbes, freqx, freqy, fwhmb2,  confac,
     *          fwhmr, g, h, hbar, lpf1, mlfilt, twopi, maxmag, arg,
     *          pi, pisq, r0, sb2dsq, sigbsq, sigesq, sigmab,
     *          sigmae, sigmar, sigmb2, sigrsq, topisq, v, w, xi,
     *          xisq, xi1, xi2, fwhme, fwhmb, r0, pix
c
      Integer i, inv, j, length, lupimg, luimg, lceof, rtncnt,
     *          n, nv2, rascos, twodim, type, rtncnt
      Integer ttin, ttout
      Integer*2 header(128)
      EQUIVALENCE (header(99), FWHME), (header(101), FWHMB)

      data pi /3.14159265/, pix /0.25/
      data ttin /5/, ttout /6/

      n = imgdmt
      nv2 = imgdmt/2
      twopi = 2.0*pi
      pisq = pi*pi
      topisq = 2.0*pisq
      confac = 2.0*sqrt(2.0*alog(2.0))

      twodim = 0
      fwhmb2 = 0.0
      xi1 = 0.0
      xi2 = 0.0
      lpf1 = 0.0

      call fopenr("Enter confidence-weighted pre-image file",lupimg)
      call fcreat("Enter output confidence-weight file     ",luimg)

      r0 = pix/2.0

      write(ttout, 5)
      read(ttin,  10) type
      write(ttout,15) type

      write(ttout,55)
      read (ttin, 35) fwhmr
      write(ttout,60) fwhmr
      sigmar = fwhmr/confac
      sigrsq = sigmar*sigmar

      twodim = 0
      IF (type .eq. 0) THEN
         write(ttout,20)
         read (ttin, 10) twodim
         write(ttout,25) twodim
      END IF
```



```
      IF (twodim .eq. 1) THEN
        write(ttout,65)
        read (ttin, 35) fwhmb2
        write(ttout,70) fwhmb2
      END IF
      sigmb2 = fwhmb2/confac
```

```
      sb2dsq = sigmb2*sigmb2

      write(ttout,75)
      read (ttin, 10) rascos
      write(ttout,80) rascos

    5 format(/,
     *        '  please input the type of reconstruction desired.     ',/,
     *        '        type = 0 ==>  confidence-weighted.             ',/,
     *        '        type = 1 ==>  most-likely-position.            ',/,
     *        '        type = 2 ==>  desired image.                   ',/,
     *        '  do not include a decimal point.                      ',/,
     *        '                                                       ',/,
     *        '  type = ?                                             ',/)
   10 format(bn,i3)
   15 format(/,'  type = ', i3)
   20 format(/,
     *        '  is it desired to form a two-dimensional confidence-',/,
     *        '  weighted array before filtering?                     ',/,
     *        '        twodim = 0 ==>  do not form the 2d cw array.   ',/,
     *        '        twodim = 1 ==>  form the 2d cw array.          ',/,
     *        '  do not include a decimal point.                      ',/,
     *        '                                                       ',/,
     *        '  twodim = ?                                           ',/)
   25 format(/,'  twodim = ', i3)
   35 format(f20.10)
   55 format(/,
     *        '  please input the full width half max form of the    ',/,
     *        '  standard deviation of the circularly symmetric       ',/,
     *        '  gaussian resolution cell, fwhmr.  include the        ',/,
     *        '  decimal point.                                       ',/,
     *        '                                                       ',/,
     *        '  fwhmr = ? cm.                                        ',/)
   60 format(/,'  fwhmr = ', f10.5)
   65 format(/,
     *        '  please input the full width half max form of the    ',/,
     *        '  standard deviation of the circularly symmetric       ',/,
     *        '  gaussian which is to be used to create the two-      ',/,
     *        '  dimensional confidence-weighted preimage array.      ',/,
     *        '  include the decimal point.                           ',/,
     *        '                                                       ',/,
     *        '  fwhmb2 = ? cm.                                       ',/)
   70 format(/,'  fwhmb2 = ', f10.5)
   75 format(/,
     *        '  is it desired to use a low pass filter of the        ',/,
     *        '  raised cosine type?                                  ',/,
     *        '        rascos = 0 ==>  do not use this lpf.           ',/,
     *        '        rascos = 1 ==>  use this lpf type.             ',/,
     *        '  do not include a decimal point.                      ',/,
     *        '                                                       ',/,
     *        '  rascos = ?                                           ',/)
   80 format(/,'  rascos = ', i3)
```



```
c        read in the preimage array arreal (arreal is in the space
c     domain).  arreal is of real type.
```

```
      call frew(lupimg)
      length = 256
      IF (type.ne.2) call fread(lupimg,header,length,lceof,rtncnt)
      length = 4*imgdmt**2
      call fread(lupimg,arreal,length,lceof,rtncnt)
      IF(lceof.eq.1) THEN
         write(ttout,*) ' EOF encountered while reading input data.'
      END IF
c
      PRINT *, 'Enter FWHME'
      READ  *, fwhme
      PRINT *, 'Enter FWHMB'
      READ  *, fwhmb
      sigmae = fwhme/confac
      sigmab = fwhmb/confac
      sigesq = sigmae*sigmae
      sigbsq = sigmab*sigmab

      IF (rascos .eq. 1) THEN
        xi2 = (sigmab/sigmar)/(2*pix)
        xi1 = 0.8*xi2
      END IF

      DO 990 i = 1, imgdmt
        DO 995 j = 1, imgdmt
           arimag(i,j) = 0.0
995     CONTINUE
990   CONTINUE

c     take the two-dimensional fourier transform of the preimage.

      inv = -1
      call fft(arreal, arimag, n, n, 1, inv)
      call fft(arreal, arimag, 1, n, n, inv)

      maxmag = 0
      DO 370 i = 1, imgdmt
        DO 371 j = 1, imgdmt
           maxmag = max(maxmag, sqrt(arreal(i,j)**2 + arimag(i,j)**2))
371     CONTINUE
370   CONTINUE

      DO 380 i = 1, imgdmt
        DO 381 j = 1, imgdmt
           arreal(i,j) = arreal (i,j)/maxmag
           arimag(i,j) = arimag (i,j)/maxmag
381     CONTINUE
380   CONTINUE

c     calculate [illegible]

      DO 1010 i = 1, imgdmt
        IF (i .le. n/2) THEN
           freqx = float(i - 1)/(float(n)*pix)
        else
```

```
        freqx = float(i - 1)/(float(n)*pix) - 1.0/pix
      END IF
      DO 1015 j = 1, imgdmt
        IF (j .le. nv2) THEN
            freqy = float(j - 1)/(float(n)*pix)
          else
            freqy = float(j - 1)/(float(n)*pix) - 1.0/pix
          END IF

          xisq = (freqx*freqx) + (freqy*freqy)
          xi = sqrt(xisq)

          IF (type .eq. 0) THEN
            v = topisq*(sigesq - sigbsq)*xisq
            cwfilt = exp(-topisq*xisq*(sigrsq
     *                 - 2.0*sigbsq))/expbes(v)
            arreal(i,j) = arreal(i,j)*cwfilt
            arimag(i,j) = arimag(i,j)*cwfilt
          END IF

          IF (type .eq. 1) THEN
            h = exp(-topisq*sigrsq*xisq)
            v = pisq*(sigesq - sigbsq)*xisq
            w = twopi*xi*r0
            IF (xi .ne. 0.0) THEN
                g = exp(-topisq*sigbsq*xisq)
     *                      *  expbes(v)*besj1(w)*r0/xi
            else
                g = exp(-topisq*sigbsq*xisq)
     *                     *expbes(v)*pi*r0*r0
            END IF
            mlfilt = pi*r0*r0*h/g
            arreal(i,j) = arreal(i,j)*mlfilt
            arimag(i,j) = arimag(i,j)*mlfilt
          END IF

          IF (type .eq. 2) THEN
            arg = -topisq*sigrsq*xisq
            IF (arg .ge. -70.0) THEN
                h = exp(arg)
            else
                h = 0.0
            END IF
            arreal(i,j) = arreal(i,j)*h
            arimag(i,j) = arimag(i,j)*h
          END IF

          IF (twodim .eq. 1) THEN
            hbar = exp(-topisq*sb2dsq*xisq)
            arreal(i,j) = arreal(i,j)*hbar
            arimag(i,j) = arimag(i,j)*hbar
          END IF

          IF (rascos .eq. 1) THEN
            IF (xi .lt. xi1) THEN
                lpf1 = 1.0
```

```
                  END IF
                  IF ((xi .ge. xi1) .and. (xi .lt. xi2)) THEN
                      lpf1 = 0.5*(1 + cos(pi*(xi - xi1)/(xi2 - xi1)))
                  END IF
                  IF (xi .ge. xi2) THEN
                      lpf1 = 0.0
                  END IF
                  arreal(i,j) = arreal(i,j)*lpf1
                  arimag(i,j) = arimag(i,j)*lpf1
                END IF
1015            CONTINUE
1010      CONTINUE

c       arreal and arimag are now the real and imaginary parts of
c       the fourier transform of the desired image.

c       take the inverse two-dimensional fourier transform to
c       obtain the desired image.

        inv = +1
        call fft(arreal, arimag, n, n, 1, inv)
        call fft(arreal, arimag, 1, n, n, inv)

c       arreal is now the desired image in the space domain.
c       save the desired image on disc.

        call frew(luimg)
        length = 256
        call fwrite(luimg, header, length)
        length = 4*imgdmt**2
        call fwrite(luimg,arreal,length)

        call fclose(lupimg)
        call fclose(luimg)
        stop ' final image written to disc '
        END
```

```
C*******************************************************************
C
C       PROGRAM TITLE       vufinl.f
C
C       WRITTEN BY          J. Trent Wohlschlaeger
C       DATE WRITTEN        Nov 10, 1986
C       WRITTEN FOR         Delay-Doppler Radar Imaging Group
C
C
C
C       PROGRAM INTENT
C              This is a graphics utility program which displays
C        the final image created by the cw algorithm.
C
C*******************************************************************

      Implicit Logical (A-Z)
      Include 'parm.com'
      Integer xoff,yoff,xsiz,ysiz
      Real arreal (imgdmt, imgdmt)
      Integer  length, luimg, lceof, rtncnt
      Integer*2 header(128)
      Integer i, j
      Real pic (128,128)

C PARAMETERS
      Parameter (xsiz =  0, ysiz =  0)

C -------------------------------------------------------------
C INITIALIZE GRAPHICS PROCESSOR
      CALL MGIASNGP (0,0)

C ENABLE ALL PLANES
      CALL MGIPLN (-1)

      CALL fopenr ("Enter confidence-weighted image file", luimg)
      CALL frew (luimg)
      length = 256
      CALL fread (luimg, header, length, lceof, rtncnt)
      length = 4*imgdmt**2
      CALL fread (luimg, arreal, length, lceof, rtncnt)
      IF (lceof .eq. 1) THEN
         PRINT *, ' End-of-file encountered while reading input data.'
      END IF
      xoff = 0
      yoff = 0

      DO 10 i = 1,128
        DO 20 j = 1, 128
           pic (i,j) = arreal (i+64, j+64)
20       CONTINUE
10    CONTINUE

      CALL DspImg (pic,128,128,xoff,yoff,xsiz,ysiz)

9999  CONTINUE
      CALL MGIDEAGP
```

```
      END
```